

University of Alberta Library



0 1620 1675 1164

QA
76
A333
1997
gr.07-12
amend.

2002
CURRGDHT



EX LIBRIS
UNIVERSITATIS
ALBERTENSIS

2002 CTS AMENDMENTS to the Information Processing Guide to Standards and Implementation

Summary of Curriculum Changes	
<ul style="list-style-type: none"> Prerequisite requirements have been removed from: <ul style="list-style-type: none"> INF2030: Keyboarding 2 INF2040: Keyboarding 3 INF3030: Keyboarding 4 INF3040: Keyboarding 5 INF3050: Keyboarding 6 	<ul style="list-style-type: none"> New Courses: <ul style="list-style-type: none"> INF1210: Computer Science 1 INF2210: Computer Science 2 INF3210: Computer Science 3 INF2220: Object-oriented Programming 1 INF3220: Object-oriented Programming 2 INF3230: Dynamic Data Structures 1 INF3240: Dynamic Data Structures 2
<ul style="list-style-type: none"> Effective September 2002, Section I has been removed from all CTS strands and replaced with a general information page. 	

Section B

- Remove** pages B.1–B.4 (1997) and **replace** with new pages B.1–B.4 (Revised 2002).
- Remove** pages B.5–B.8 (1999) and **replace** with new pages B.5–B.10 (Revised 2002).

Section D

- Remove** pages D.1–D.2 (1997) and **replace** with new pages D.1–D.2 (Revised 2002).
- Remove** pages D.41–D.42 (Revised 1999) and **replace** with new pages D.41–D.42 (Revised 2002).
- Add** new pages D.45–D.50 (2002).

Section E

- Remove** pages E.1–E.2 (1997) and **replace** with new pages E.1–E.2 (Revised 2002).
- Remove** pages E.9–E.10 (1997) and **replace** with new pages E.9–E.10 (Revised 2002).
- Remove** pages E.13–E.14 (1997) and **replace** with new pages E.13–E.14 (Revised 2002).
- Add** new pages E.97–E.108 (2002).

Section F

- Remove** pages F.1–F.2 (1997) and **replace** with new pages F.1–F.2 (Revised 2002).
- Remove** pages F.13–F.14 (1997) and **replace** with new pages F.13–F.14 (Revised 2002).
- Remove** pages F.17–F.18 (1997) and **replace** with new pages F.17–F.18 (Revised 2002).
- Remove** pages F.21–F.22 (1997) and **replace** with new pages F.21–F.22 (Revised 2002).
- Add** new pages F.87–F.106 (2002).

Section G

- Remove** pages G.3–G.4 (1997) and **replace** with new pages G.3–G.4 (Revised 2002).
- Add** new pages G.59–G.80 (2002).

Section I

- Remove** Section I (Revised 2000) and **replace** with new page I.1 (Revised 2002).

INFORMATION PROCESSING

B. STRAND RATIONALE AND PHILOSOPHY

Information Processing, a strand in Career and Technology Studies, represents the study of electronic technologies as they apply to personal use and the business environment.

As we move more rapidly into the information age, it is crucial that students are able to use electronic technologies to access and manipulate information in an efficient manner. Accurate, timely information is the basis for sound decision making and effective communication.

As students build confidence in their understanding of the various information processing tools and procedures, they will be able to transfer their knowledge and skill to a wide range of contexts. They will also be better able to adapt to the continual changes caused by the evolving technologies.

To understand the shift from the *industrial society* toward the *information age*, it is important that a student understands the significance of the current technological development, of how technology affects an individual's daily life and of the impact that technology has on the world of work. Within this perspective, Information Processing provides for the development of:

- an understanding of the systems that relate in whole or in part to the management of information
 - an understanding of the ethical and societal issues concerning technological development and its impact on society
 - technological skills and knowledge designed for personal use
 - technological skills and knowledge that transfer to other curriculum areas
 - technological skills and knowledge required for the world of work.
- Students will learn to input, process and output information in the following areas:
- system operations
 - text/data input
 - productivity software
 - applied processing
 - dynamic environment
 - programming (procedure-oriented and object-oriented)
 - computer science.
- a meaningful study of technological trends

STRAND ORGANIZATION

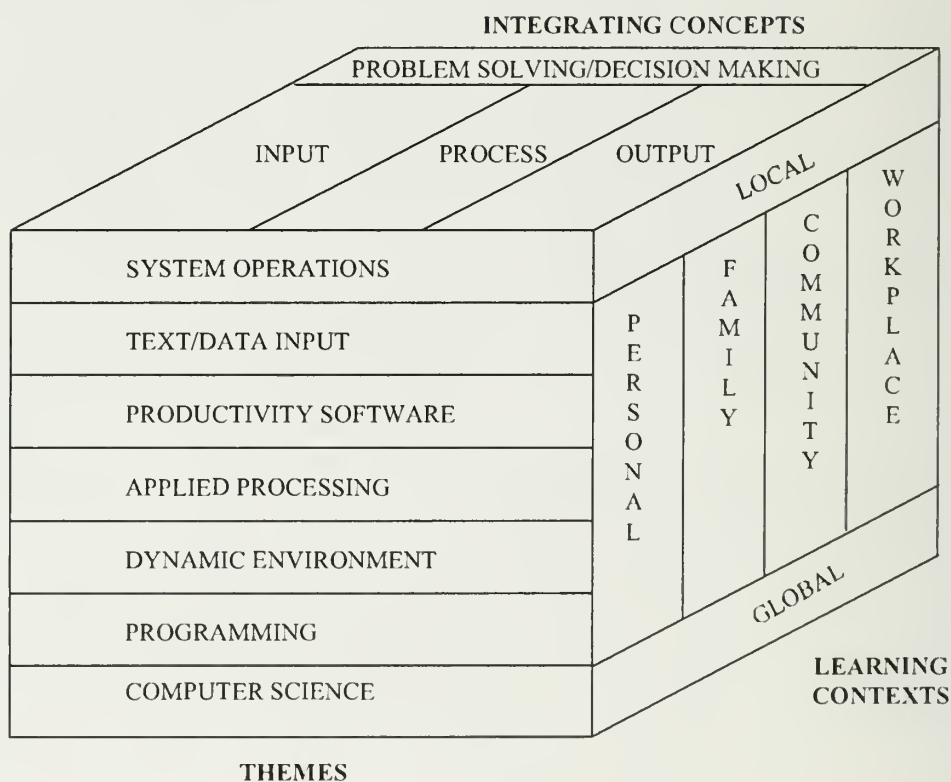
The developmental model indicates the relationship of what the students learn (as described in the themes), how these learnings are emphasized within the courses (as described in the integrating concepts) and how students will apply these learnings (as described in the learning contexts).

LEVELS

Students working on courses at the introductory level develop basic techniques and skills which, while primarily for personal use, also form the foundation for the development of more professional applications.

In the intermediate level courses, students are expected to work more independently and expand and refine basic skills in a wide range of applications.

At the advanced level, students use initiative to efficiently integrate applications and processes to produce high quality work to workplace standards.



THEMES

The themes provide learning experiences that link knowledge, skills and attitudes with real-life situations. Courses are organized into seven themes:

- system operations
- text/data input
- productivity software
- applied processing
- dynamic environment
- programming
- computer science.

The courses in the System Operations theme help students efficiently use and assess computer hardware and related software and peripherals, and understand and apply various communication protocols.

In the Text/Data Input themes, students develop efficient keyboarding competencies for both personal use and professional skill levels.

In Productivity Software courses, students learn the commands and processes of the key productivity software packages used in personal and professional applications, including word processing, spreadsheet, database, graphics and electronic/desktop publishing. Students expand their ability to use these software applications in other CTS strands such as Communication Technology, or in other courses such as English language arts, mathematics.

The Applied Processing theme is designed to increase students' level of productivity as they produce a variety of documents that integrate text, data and graphics applications.

In the Dynamic Environment theme, students work with software that links various media and processes in new and unique ways to manage and communicate information.

The Programming theme provides an opportunity for students to develop high-level, structured programming skills, using either procedure-oriented or object-oriented processes.

The Computer Science theme provides opportunities for students to develop skills in a high growth career area of the emerging economy. Courses within this theme address a need identified by business/industry and post-secondary institutes for:

- more senior high school students to have access to training in computer science
- more consistent standards for literacy in computer science
- smoother transitions from secondary to post-secondary education.

INTEGRATING CONCEPTS

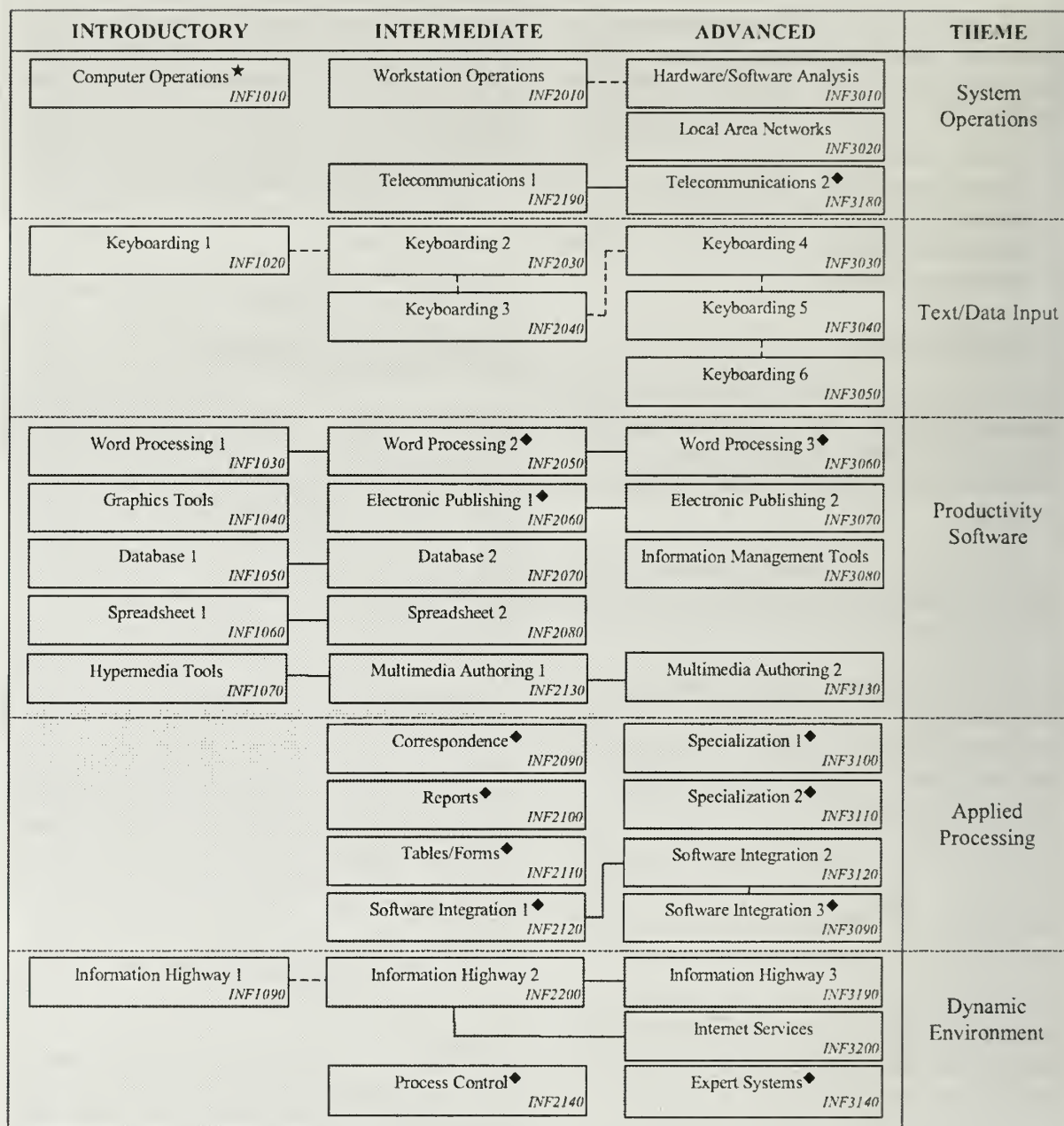
Integrated within each of the Information Processing courses is the expectation that students will identify and resolve problems efficiently by using effective decision-making skills. Students apply these problem-solving/decision-making skills as they determine the most effective and efficient processes to use to input, process and output information.

LEARNING CONTEXTS

Learning contexts help students relate their learning to real-life experiences and challenges. In courses at the introductory level, these challenges are most frequently in a context typical in daily living—within the home, school or community. As the student progresses through the intermediate and advanced levels, the challenges and related expectations for performance involve contexts that relate to the workplace.

With the ever-increasing power of information technologies, all of these applications can be applied both at the local and global level. The competencies students develop in Information Processing will also support students as they continue their education in post-secondary or other further education opportunities.

SCOPE AND SEQUENCE



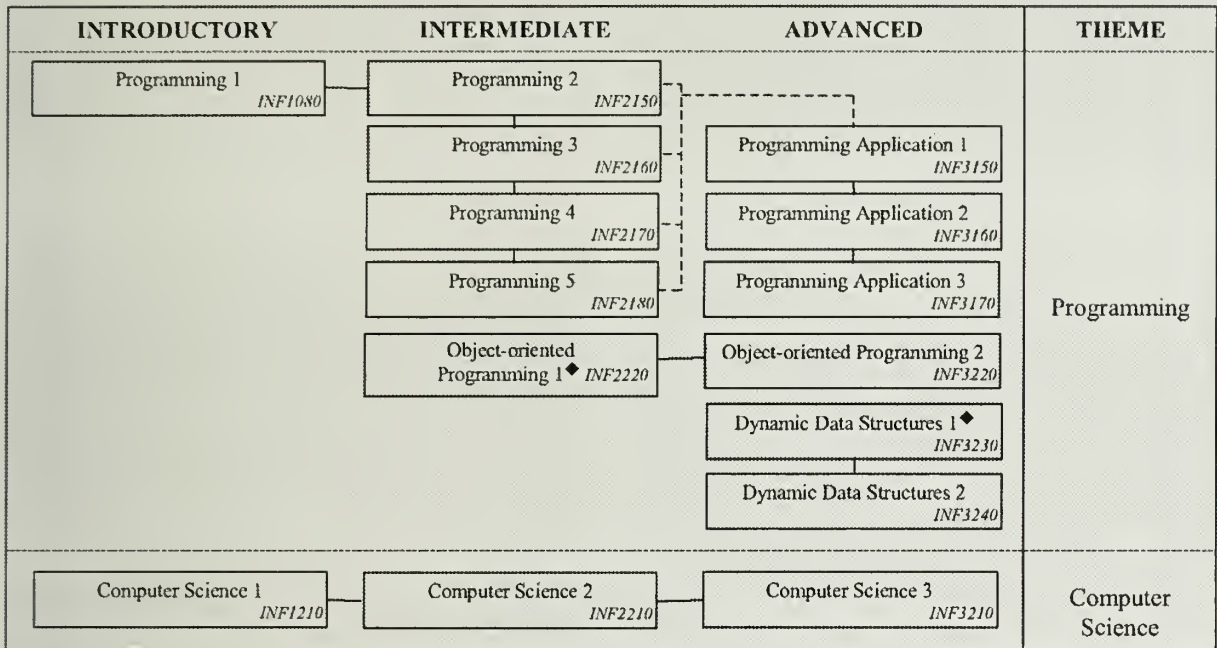
—— Prerequisite

----- Recommended sequence

★ Course provides a strong foundation for further learning in this strand.

♦ Refer to specific courses for additional prerequisites.

SCOPE AND SEQUENCE (continued)



_____ Prerequisite - - - - - Recommended sequence

♦ Refer to specific courses for additional prerequisites.

COURSE DESCRIPTIONS

Course INF1010: Computer Operations

Students develop personal use skills basic to all courses in the Information Processing strand in the following applications: file management, basic hardware and software operations, text entry and workstation routines.

Course INF1020: Keyboarding 1

Students develop accurate touch keystroking of text and data appropriate to personal use and the application of efficient workstation procedures.

Course INF1030: Word Processing 1

Students develop skill in using basic commands and functions in word processing software, including document editing, and the formatting and printing of reports, correspondence and tables suitable for personal use applications.

Course INF1040: Graphics Tools

Students learn the basic commands and functions of computer graphics software, including bitmapped graphics (paint program) and vector graphics (draw program). Students also develop basic skills in manipulating existing graphics, as well as in producing their own graphics.

Course INF1050: Database 1

Students are introduced to the basic commands and functions of database software, and demonstrate how this software can be used as a personal tool in data and information management.

Course INF1060: Spreadsheet 1

Students have an opportunity to use basic functions and commands in spreadsheet software for general data manipulation and personal record keeping.

Course INF1070: Hypermedia Tools

Students develop basic skills with tools used for computerized presentations involving text, data, graphics, sound and animation.

Course INF1080: Programming 1

Students are introduced to computer programming languages and a structured programming environment, and they construct algorithms and code instructions to solve identified problems.

Course INF1090: Information Highway 1

Students develop personal use Internet skills for accessing and communicating data and information, with particular emphasis on the world wide web and email.

Course INF1210: Computer Science 1

Students are introduced to the nature, approaches and areas of interest of computer science and its relationship to areas, such as computer engineering and information technology. Students explore concepts associated with hardware, software and processes at an introductory level. There is an emphasis on sequential and structured programming approaches.

Course INF2010: Workstation Operations

Students learn computer workstation operations, including computer architecture, peripherals, configurations, operating system environments and platforms, utility software, diagnostic and protection software, hard drive file updating and maintenance, support resource application and troubleshooting activities.

Course INF2030: Keyboarding 2

Students enhance their personal use keyboarding competencies by increasing the rate of accurate touch keystroking of the alphabetic, numeric and selected punctuation keys.

Course INF2040: Keyboarding 3

Students enhance their keyboarding competencies, by increasing the rate of accurate touch keystroking of alphabetic, numeric and all punctuation keys to support personal use and limited, entry-level, workplace opportunities.

Course INF2050: Word Processing 2

Students expand their skills in using word processing software commands and functions to produce mailable reports and correspondence, including letters, memorandums and tables, all from rough draft copy.

Course INF2060: Electronic Publishing 1

Students develop skill, using electronic/desktop publishing software to create a variety of camera-ready documents.

Course INF2070: Database 2

Students use all the commands and functions of electronic database software that support effective and efficient database applications.

Course INF2080: Spreadsheet 2

Students demonstrate advanced level spreadsheet commands and functions to calculate and manipulate data and to prepare appropriate reports and printouts in text and graphic format.

Course INF2090: Correspondence

Students expand their rate of document production as they prepare various forms of correspondence in mailable form, using word processing software.

Course INF2100: Reports

Students expand their rate of production as they prepare various reports and manuscripts in mailable form.

Course INF2110: Tables/Forms

Students expand their rate of document production as they prepare various tables/forms in mailable form.

Course INF2120: Software Integration 1

Students develop document production skills requiring the integration of data, text and graphics.

Course INF2130: Multimedia Authoring 1

Students are introduced to multimedia software and provided with an opportunity to develop basic authoring competence, by accessing and integrating software resident text, video and audio clips.

Course INF2140: Process Control

Students develop skills in robotics/simulation software control by creating, modifying and using programs that incorporate computer-controlled movements and events in robotics/simulation activities and applications.

Course INF2150: Programming 2

Students increase their programming skills, by designing and generating programming code to handle decision making and repetitive processes.

Course INF2160: Programming 3

Students increase their programming skills, by using subprogram structures.

Course INF2170: Programming 4

Students increase their programming skills, by developing and using derived data types.

Course INF2180: Programming 5

Students increase their programming skills, by developing and using recursive, sorting and merging algorithms.

Course INF2190 Telecommunications 1

Students learn how to select and use various wired and wireless telecommunication systems. By using the Internet, they investigate how communication principles, bandwidth, telecommunication infrastructure and wave spectrum affects telecommunication systems.

Course INF2200: Information Highway 2

Students learn how to produce a web page for the Internet.

Course INF2220: Object-oriented Programming 1

Students are introduced to object-based programming (OBP) and object-oriented programming (OOP). They develop algorithms, using introductory object-oriented design techniques, and use these algorithms to write introductory object-based and object-oriented programs.

Course INF2210: Computer Science 2

Students extend their knowledge of the discipline of computer science by exploring the modular paradigm and its impact on algorithm development and implementation (programming). Students also add to their understanding by exploring the stylized von Neumann computer system at the machine level, and by examining the impact of computer science and computer technology on society.

Course INF3010: Hardware/Software Analysis

Students analyze, compare and evaluate hardware/software based on user requirements.

Course INF3020: Local Area Networks

Students learn about local area network (LAN) computer systems, including hardware and peripheral configurations, interface protocols and data transmission characteristics.

Course INF3030: Keyboarding 4

Students develop their text and data keyboarding skills to entry-level occupational expectations.

Course INF3040: Keyboarding 5

Students increase their occupational-level keyboarding competence of text, data and function/service keys, using straight copy and edited material.

Course INF3050: Keyboarding 6

Students enhance their occupational-level keyboarding competence of all keystroke functions, using unedited, edited and straight copy material.

Course INF3060: Word Processing 3

Students develop occupational-level competence in the use of word processing software commands and functions to produce mailable reports, correspondence and tables, including the importing and merging of text, data and graphics.

Course INF3070: Electronic Publishing 2

Students use the functions and commands of electronic/desktop publishing software as they integrate text composing, editing, typesetting, graphics generation and page layout functions to create customized, professional, quality documents.

Course INF3080: Information Management Tools

Students develop competence in using information management systems software, such as project management, schedules and planners for either personal or workplace applications.

Course INF3090: Software Integration 3

Students develop high production rates as they process documents from unedited and unformatted copy, using numerous functions/commands to create, revise, format and print a wide range of mailable copy.

Course INF3100: Specialization 1

Students specialize in document preparation, terminology application and associated office routine expectations in a specific focus area, such as a medical, legal, petroleum, real estate, insurance, travel/tourism, forestry or agricultural environment.

Course INF3110: Specialization 2

Students develop workplace competence in a specific focus area, such as medical, legal, petroleum, real estate, insurance, travel/tourism, forestry or agricultural environment, by creating and completing appropriate documents that employ specialized communication skills and conform to workplace expectations and time constraints.

Course INF3120: Software Integration 2

Students expand their document production skills to workplace standards. Documents could require the importing and integration of word processing, spreadsheet, graphics and database files.

Course INF3130: Multimedia Authoring 2

Students learn to use a multimedia file or multimedia authoring software based on digitized input of text, video and audio clips.

Course INF3140: Expert Systems

Students acquire knowledge of expert systems, such as artificial intelligence and virtual reality. They gain competence, by developing or modifying programs that incorporate computer-controlled environments and multimedia interactive activities and applications.

Course INF3150: Programming Application 1

Students create programs that use external files.

Course INF3160: Programming Application 2

Students create a program, using a second programming language.

Course INF3170: Programming Application 3

Students enhance a program, using a second programming language.

Course INF3180: Telecommunications 2

Students demonstrate knowledge of telecommunication systems by designing a new system. They use the Internet in researching and developing their design and for comparing and contrasting various telecommunication initiatives. Students analyze the effect this is having on the individual and society.

Course INF3190: Information Highway 3

Students develop and maintain an Internet/intranet web site that makes use of advanced features.

Course INF3200: Internet Services

Students expand their skills from Information Highway 2, by learning how to operate, maintain and build an Internet/intranet site that may include computer bulletin boards, forums, electronic mail, Internet list servers, and/or moderated newsgroups. Proper use of hardware, software and liaison with users and clients is emphasized.

Course INF3220: Object-oriented Programming 2

Students extend their knowledge of object-oriented programming (OOP) concepts. They increase their expertise in object-oriented design and programming by developing algorithms and programs that use templated classes, containment and inheritance to promote reusability.

Course INF3230: Dynamic Data Structures 1

Students are formally introduced to dynamic data structures in general and to linked lists in particular.

Course INF3240: Dynamic Data Structures 2

Students add to their understanding of dynamic data structures by developing introductory algorithms and programs that use stacks, queues and trees.

Course INF3210: Computer Science 3

Students extend their knowledge of the core concepts of the discipline of computer science by exploring more advanced aspects of the modular programming paradigm and by beginning their examination of the object-oriented programming paradigm. Students also add to their understanding by manipulating a Turing machine and by analyzing the nature of the emerging information society.

COURSE CURRICULUM AND ASSESSMENT STANDARDS:

INTRODUCTORY LEVEL

The following pages define the curriculum and assessment standards for the introductory level of Information Processing.

Introductory level courses help students build daily living skills and form the basis for further learning. Introductory courses are developed for students who have no previous experience in the strand.

General outcomes define the competencies a student must demonstrate to achieve success in a course. Assessment standards define the conditions and criteria to be used for assessing the competencies defined in the course learner expectations.

Specific outcomes provide a detailed framework for instruction to help students build the competencies defined in the general outcomes. Additional information and suggestions for instruction are provided in the Notes column; teachers may wish to use this space to record their ideas for instruction or student projects.

Course INF1010:	Computer Operations	D.3
Course INF1020:	Keyboarding 1	D.7
Course INF1030:	Word Processing 1	D.11
Course INF1040:	Graphics Tools	D.15
Course INF1050:	Database 1	D.19
Course INF1060:	Spreadsheet 1	D.25
Course INF1070:	Hypermedia Tools	D.31
Course INF1080:	Programming 1	D.35
Course INF1090:	Information Highway 1	D.41
Course INF1210:	Computer Science 1	D.45

COURSE INF1090: INFORMATION HIGHWAY 1**Level:** Introductory**Theme:** Dynamic Environment**Prerequisite:** None**Description:** Students develop personal use Internet skills for accessing and communicating data and information, with particular emphasis on the Internet and e-mail.**Parameters:** Access to a computer workstation and the Internet.**Supporting Course:** INF1030 Word Processing I**Curriculum and Assessment Standards**

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<i>The student will:</i> <ul style="list-style-type: none">demonstrate knowledge of the history of the Internet and of its basic functions	<i>Assessment of student achievement should be based on:</i> <ul style="list-style-type: none">a project related to:<ul style="list-style-type: none">history of the Internetaccess to Internetusing basic terminology and commandsexploring the Internet to discover its potentialfinding information regarding proper "netiquette" (Internet etiquette)personal safety and security. <i>Assessment Tool</i> <i>Assessment Guide: Information Highway 1 – Getting Started (INF1090-1)</i> <i>Standard</i> <i>Rating of 1 for each applicable task</i>	20
<ul style="list-style-type: none">demonstrate ability to communicate with others through the Internet	<ul style="list-style-type: none">communicating through the Internet (internal or external) using e-mail and at least one other of the following:<ul style="list-style-type: none">on-line chattingnewsgroupsmailing lists/listservsother technologies as they emerge. <i>(Note: This is a dynamic list that changes rapidly as technologies come and go; learning opportunities should reflect what is currently available.)</i> <i>Assessment Tool</i> <i>Assessment Guide: Information Highway 1 – Communicating (INF1090-1)</i> <i>Standard</i> <i>Rating of 1 for each applicable task</i>	30

COURSE INF1090: INFORMATION HIGHWAY 1 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> demonstrate ability to access and report specific information from the Internet 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> accessing specific information through a prescribed research topic: <ul style="list-style-type: none"> use a variety of directories and search engines to locate specific information download information cut/paste/edit, format collected data into a report/presentation properly cite information from Internet sources. <p><i>Assessment Tool</i> <i>Assessment Guide: Information Highway 1 – Access and Report Specific Information (INF1090-1)</i></p> <p><i>Standard</i> <i>Rating of 1 for each applicable task</i></p>	40
<ul style="list-style-type: none"> apply, consistently, appropriate workstation routines 	<ul style="list-style-type: none"> demonstration of appropriate workstation routines. <p><i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i></p> <p><i>Standard</i> <i>Rating of:</i> 1 – Workstation Use 2 – File Management 1 – Time Management/Organization 2 – Professionalism</p>	10
<ul style="list-style-type: none"> demonstrate basic competencies. 	<ul style="list-style-type: none"> observation of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	Integrated throughout

COURSE INF1210: COMPUTER SCIENCE 1**Level:** Introductory**Theme:** Computer Science**Prerequisite:** None

Description: Students are introduced to the nature, approaches and areas of interest of computer science and its relationship to areas, such as computer engineering and information technology. Students explore concepts associated with hardware, software and processes at an introductory level. There is an emphasis on sequential and structured programming approaches.

Parameters: Designed to be taught in conjunction with INF1080 Programming 1, INF2150 Programming 2, INF1090 Information Highway 1 and INF1070 Hypermedia Tools as a Grade 10 course in Computer Science.

Curriculum and Assessment Standards

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<i>The student will:</i> <ul style="list-style-type: none">identify and describe the nature, approaches and areas of interest of computer science	<i>Assessment of student achievement should be based on:</i> <ul style="list-style-type: none">a test, presentation or project designed to address the following topics:<ul style="list-style-type: none">computer science's central focus on the nature and techniques of problem solvingthe role of the algorithm as a foundation of the discipline of computer sciencethe general areas of interest of computer sciencethe relationship among computer science, computer engineering and information technologycommon misconceptions about computer science	10
<ul style="list-style-type: none">explain and demonstrate the nature, developmental process, use of basic algorithms associated with input processing output (IPO) and structured approaches, and application of these idioms to create complex algorithms	<ul style="list-style-type: none">a presentation or project designed to demonstrate:<ul style="list-style-type: none">the basic nature of algorithmsthe ability to design, develop and explain IPO (sequential) and structured algorithmsproficient use of key basic algorithms (idioms) and the ability to use these idioms to create other, complex algorithms	20

COURSE INF1210: COMPUTER SCIENCE 1 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> explain and demonstrate the nature, evolution, types and role of programming languages 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> a presentation or project designed to explain and demonstrate: <ul style="list-style-type: none"> the basic nature, evolution, types and role of programming languages 	20
<ul style="list-style-type: none"> explain and demonstrate the rationale, three fundamental control structures and representation of data in sequential and structured programs 	<ul style="list-style-type: none"> a presentation or project demonstrating: <ul style="list-style-type: none"> the translation of algorithms into structured programs the basic nature, approach and representation of data in sequential and structured programs 	30
<ul style="list-style-type: none"> explain the nature, evolution and basic architecture of the von Neumann computer system 	<ul style="list-style-type: none"> a presentation or project addressing the evolution and nature of the von Neumann computer architecture under the direction of a simple program to explain: <ul style="list-style-type: none"> the nature of the five main hardware “blocks” of the computer a number of typical devices that make up each block the flow of data through each block of the computer the relationship with the basic data-processing paradigm. <p><i>Assessment Tool</i> <i>Assessment Checklist: Computer Science 1 Concepts (Introductory), INF1210-1</i> <i>Sample Assignments: Computer Science 1 Concepts (Introductory), INF1210-2</i></p> <p><i>Standard</i> <i>Rating of 2 for the Problem-solving Phase and a rating of 3 for the Implementation Phase</i></p>	20
<ul style="list-style-type: none"> demonstrate basic competencies. 	<ul style="list-style-type: none"> observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	Integrated throughout

COURSE INF1210: COMPUTER SCIENCE 1 (continued)

Concept	Specific Outcomes	Notes
Nature of Computer Science	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • discuss the following topics: <ul style="list-style-type: none"> – computer science's central focus is the study of the nature and techniques of problem solving with a particular interest in problems that are solvable by computation – the algorithmic approach is used to solve problems – computer systems are used to test/implement algorithmic solutions to problems – algorithms are used to develop generalized applications useful for solving classes of problems • describe the general areas of interest of computer science. They include: <ul style="list-style-type: none"> – development and analysis of algorithms – computing systems and their components – communication—both human/machine and machine/machine – formal languages—natural and artificial – automata – artificial intelligence – general development of IT applications • compare and explain computer science versus computer engineering and information technology <ul style="list-style-type: none"> – theoretical versus applied – general versus specific – exploratory versus applicatory • describe some of the misconceptions associated with computer science: <ul style="list-style-type: none"> – is the study of computer systems – is synonymous with programming – is the learning of various computer applications 	

COURSE INF1210: COMPUTER SCIENCE 1 (continued)

Concept	Specific Outcomes	Notes
Algorithmic Problem Solving	<p><i>The student should:</i></p> <ul style="list-style-type: none"> describe how an algorithm: <ul style="list-style-type: none"> is a step-by-step set of instructions that results in a solution to a problem becomes a computer program when expressed in a programming language demonstrate iterative and incremental approaches in the analysis and design stages of the software development process carry out the first two steps of the Systems Development Life Cycle (Analysis and Design) using: <ul style="list-style-type: none"> flowcharts pseudocode IPO charting demonstrate a number of core algorithms, such as: <ul style="list-style-type: none"> accumulation (keeping a running total) determining the mean determining minimums and maximums 	
Implementing the Algorithm (Software and Software Development)	<ul style="list-style-type: none"> demonstrate the third step of the Systems Development Life Cycle (Development or Coding) using iterative and incremental approaches demonstrate the nature of programming language; specifically, that these languages: <ul style="list-style-type: none"> reflected a simplified version of natural language <ul style="list-style-type: none"> grammar syntax semantics imperative vocabulary statements blocks evolved in tandem with algorithms and hardware through 5 “generations” a continuum from machine language to natural language programming 	

COURSE INF1210: COMPUTER SCIENCE 1 (continued)

Concept	Specific Outcomes	Notes
	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • each successive generation closer to natural or human language • each generation requires more sophisticated translation into a machine understandable form (assemblers, compilers, interpreters) • “higher” generation languages easier for humans to use but slower and less machine efficient • first generation: machine language • second generation: assembly language • third generation: high-level languages • fourth generation: computer-assisted programming languages • fifth generation: natural language programming – reflected IPO or the data processing paradigm <ul style="list-style-type: none"> • initialization • input statements • processing statements • output statements • termination/linking • demonstrate how programming languages dealt with data representation: <ul style="list-style-type: none"> – binary and hexadecimal systems – standard data types – data storage • demonstrate structured programming concepts: <ul style="list-style-type: none"> – rationale for structured programming – goto-less programming – three fundamental control structures (sequential, decision and iterative) • demonstrate iterative and incremental approaches in the Implementation and Maintenance stages of the Systems Development Life Cycle 	

COURSE INF1210: COMPUTER SCIENCE 1 (continued)

Concept	Specific Outcomes	Notes
Executing the Algorithm (Computer Systems)	<p><i>The student should:</i></p> <ul style="list-style-type: none">• demonstrate computer architecture by producing/explaining/describing:<ul style="list-style-type: none">– a block diagram of a stereotypical von Neumann machine<ul style="list-style-type: none">• input block or stage• processing block or stage• output block or stage• internal storage block or stage (memory)• external storage block or stage (memory)– a number of typical devices associated with each “block”<ul style="list-style-type: none">• i.e., keyboard or mouse with the input block– a flow of data through the computer under the direction of a program.	

COURSE CURRICULUM AND ASSESSMENT STANDARDS: SECTION E: INTERMEDIATE LEVEL

The following pages define the curriculum and assessment standards for the intermediate level of Information Processing.

Intermediate level courses help students build on the competencies developed at the introductory level and focus on developing more complex competencies. They provide a broader perspective, helping students recognize the wide range of related career opportunities available within the strand.

Course INF2010:	Workstation Operations	E.3
Course INF2030:	Keyboarding 2.....	E.9
Course INF2040:	Keyboarding 3.....	E.13
Course INF2050:	Word Processing 2.....	E.17
Course INF2060:	Electronic Publishing 1	E.21
Course INF2070:	Database 2	E.27
Course INF2080:	Spreadsheet 2	E.31
Course INF2090:	Correspondence	E.35
Course INF2100:	Reports	E.39
Course INF2110:	Tables/Forms	E.43
Course INF2120:	Software Integration 1	E.49
Course INF2130:	Multimedia Authoring 1	E.53
Course INF2140:	Process Control	E.57
Course INF2150:	Programming 2	E.61
Course INF2160:	Programming 3	E.67
Course INF2170:	Programming 4	E.75
Course INF2180:	Programming 5	E.81
Course INF2190:	Telecommunications 1	E.87
Course INF2200:	Information Highway 2	E.93
Course INF2220:	Object-oriented Programming 1	E.97
Course INF2210:	Computer Science 2	E.103

COURSE INF2030: KEYBOARDING 2**Level:** Intermediate**Theme:** Text/Data Input**Prerequisite:** None

Description: Students enhance their personal use keyboarding competencies by increasing the rate of accurate touch keystroking of the alphabetic, numeric and selected punctuation keys.

Parameters: Computer workstation, disk, word processing software, support resources.

Curriculum and Assessment Standards

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> demonstrate keyboarding competence: <ul style="list-style-type: none"> at 30 words per minute (wpm) numeric entry at 100 keystrokes per minute (kpm) technique 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> three timed writings, each from different straight copy material, over a period of no more than five consecutive class periods, which demonstrates proper touch keyboarding : <ul style="list-style-type: none"> on alphabetic keys <ul style="list-style-type: none"> two-minute duration maximum one uncorrected error $SI \leq 1.25$ minimum keystroke rate: 30 words per minute on numeric keys: <ul style="list-style-type: none"> one-minute duration maximum one uncorrected error 100 numeric keystrokes a minute on 1 to 3 digit numbers. <p><i>Assessment Tool</i> <i>Reference Chart: Keyboarding and Numberpad Rates (INFKEYNB)</i></p> <ul style="list-style-type: none"> observations over the last quarter of the learning period, during timings and drill work. <p><i>Assessment Tool</i> <i>Assessment Checklist: Text-Data Entry (INFTDENT)</i></p> <p><i>Standard</i> <i>Rating of:</i> 3 – Eye Focus 3 – Keystroking 2 – Service Keys 3 – Body Position</p> 	<p>50</p> <p>10</p> <p>30</p>

COURSE INF2030: KEYBOARDING 2 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • apply, consistently, appropriate workstation routines • demonstrate basic competencies. 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • demonstrating appropriate workstation routines. <p><i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i></p> <p><i>Standard</i> <i>Rating of:</i> 2 – Workstation Use 3 – File Management 2 – Time Management/Organization 3 – Professionalism</p> <ul style="list-style-type: none"> • observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	<p>10</p> <p>Integrated throughout</p>

Concept	Specific Outcomes	Notes
Text Entry	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • demonstrate increasingly rapid, accurate touch keystroking on straight copy of: <ul style="list-style-type: none"> – alphabetic keys – number keys – punctuation keys (.,:;?“()!-_) – symbol keys \$., &, % – service keys (enter, shift, delete, backspace, tab) • use function and cursor movement keys efficiently 	<p>Develop speed and accuracy at the phrase, sentence and short paragraph level using short, repetitive timings (12 seconds to one minute) with straight copy text of varying SI (1.0–1.4).</p>

COURSE INF2040: KEYBOARDING 3**Level:** Intermediate**Theme:** Text/Data Input**Prerequisite:** None

Description: Students enhance their keyboarding competencies, by increasing the rate of accurate touch keystroking of alphabetic, numeric and all punctuation keys to support personal use and limited, entry-level, workplace opportunities.

Parameters: Computer workstation, disk, word processing software, support resources.

Curriculum and Assessment Standards

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • demonstrate keyboarding competence: <ul style="list-style-type: none"> – text entry at 40 words per minute (wpm) – numeric entry at 120 keystrokes per minute (kpm) – technique 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • three timed writings, each from different straight copy material, over a period of no more than five consecutive class periods, which demonstrates proper touch keyboarding : <ul style="list-style-type: none"> – on alphabetic keys <ul style="list-style-type: none"> • two-minute duration • maximum one uncorrected error • SI 1.2 – 1.35 • minimum keystroke rate: 40 words per minute – on numeric keys: <ul style="list-style-type: none"> • one-minute duration • maximum one uncorrected error • 120 numeric keystrokes a minute on 1 to 4 digit numbers. <p><i>Assessment Tool</i> <i>Reference Chart: Keyboarding and Numberpad Rates (INFKEYNB)</i></p> <ul style="list-style-type: none"> – observations over the last quarter of the learning period, during timings and drill work. <p><i>Assessment Tool</i> <i>Assessment Checklist: Text-Data Entry (INFTDENT)</i></p> <p><i>Standard</i> <i>Rating of:</i> <i>4 – Eye Focus</i> <i>3 – Keystroking</i> <i>2 – Service Keys</i> <i>3 – Body Position</i></p>	<p>50</p> <p>10</p> <p>30</p>

COURSE INF2040: KEYBOARDING 3 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • apply, consistently, appropriate workstation routines • demonstrate basic competencies. 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • demonstrating appropriate workstation routines. <p><i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i></p> <p><i>Standard</i> <i>Rating of:</i> 2 – Workstation Use 3 – File Management 2 – Time Management/Organization 3 – Professionalism</p> <ul style="list-style-type: none"> • observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	<p>10</p> <p>Integrated throughout</p>

Concept	Specific Outcomes	Notes
Text Entry	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • demonstrate increasingly rapid, accurate touch keystroking on straight copy of: <ul style="list-style-type: none"> – alphanumeric keys – all punctuation keys – service keys (enter, shift, backspace, tab) • use function and cursor movement keys efficiently • demonstrate correct keystroking technique: <ul style="list-style-type: none"> – enter text using designated fingers – maintain home-row anchor position – demonstrate correct posture (hands, arms body) • proofread and edit text (screen and hard copy) to ensure text is error free 	<p>Develop speed and accuracy at the phrase, sentence and short paragraph level using short, repetitive timings (.5 to one minute) with straight copy text of varying SI (1.2–1.5).</p>

COURSE INF2220: OBJECT-ORIENTED PROGRAMMING 1

Level: Intermediate

Theme: Programming

Prerequisite: INF2170 Programming 4

Description: Students are introduced to object-based programming (OBP) and object-oriented programming (OOP). They develop algorithms, using introductory object-oriented design techniques, and use these algorithms to write introductory object-based and object-oriented programs.

Parameters: Access to appropriate computer equipment and software. Specifically, students must have access to an OOP environment that allows for a formal treatment of objects. They should also have access to appropriate class libraries created by other programmers.

Curriculum and Assessment Standards

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<i>The student will:</i> <ul style="list-style-type: none">• identify/describe fundamental concepts of object-oriented programming (OOP), including:<ul style="list-style-type: none">– classes, objects, member functions and instantiation– public and private access modifiers– data encapsulation– class libraries	<i>Assessment of student achievement should be based on:</i> <ul style="list-style-type: none">• a teacher-directed evaluation designed to test the student's ability to:<ul style="list-style-type: none">– describe the basic features of OOP including the use of constructor and operator overloading to add power– explain key differences between OOP and procedure-oriented programming– illustrate how OOP promotes real-world modelling, data integrity, reliability, maintainability and reusability.	10%

COURSE INF2220: OBJECT-ORIENTED PROGRAMMING 1 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> demonstrate evolving programming expertise in basic object-oriented programming, by: <ul style="list-style-type: none"> analyzing/revising/creating algorithms based on introductory object-oriented design techniques that use predefined classes to solve problems analyzing/revising/creating object-based programs, using predefined classes analyzing/revising/creating algorithms based on object-oriented design techniques that use programmer-created classes to solve problems analyzing/revising/creating object-oriented programs, using programmer-created classes 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> a series of student-developed programs that demonstrate the efficient use of object-based programming and object-oriented programming algorithms and language syntax. <p>These algorithms and programs should demonstrate the ability to:</p> <ul style="list-style-type: none"> analyze problems to determine components best represented using built-in class libraries—compiler provided libraries—and class libraries provided by other programmers revise/create algorithms that use built-in classes and class libraries use constructor and operator overloading revise/construct object-based programs that create and use appropriate instances of predefined classes analyze problems to determine components best represented using programmer-created classes and objects <ul style="list-style-type: none"> revise/create algorithms that use programmer-created classes, objects and class libraries revise/construct object-oriented programs that: <ul style="list-style-type: none"> modify/create and use simple classes modify/create and use class libraries. <p><i>Assessment Tool</i> <i>Assessment Checklist: Object-oriented Programming 1, INF2220-1</i> <i>Sample Assignments: Object-oriented Programming 1, INF2220-2</i></p> <p><i>Standard</i> <i>Rating of 2 for the Problem-solving Phase and a rating of 3 for the Implementation Phase</i></p>	<p>10%</p> <p>20%</p> <p>20%</p> <p>30%</p>

COURSE INF2220: OBJECT-ORIENTED PROGRAMMING 1 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • apply, consistently, appropriate workstation routines 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • demonstrations of appropriate workstation routines. <i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i> <i>Standard</i> <i>Rating of:</i> 2 – Workstation Routines 3 – File Management 2 – Time Management/Organization 3 – Professionalism 	10%
<ul style="list-style-type: none"> • demonstrate basic competencies. 	<ul style="list-style-type: none"> • observations of individual effort and interpersonal interaction during the learning process. <i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i> 	Integrated throughout

COURSE INF2220: OBJECT-ORIENTED PROGRAMMING 1 (continued)

Concept	Specific Outcomes	Notes
Nature of Object-oriented Programming (OOP)	<p><i>The student should:</i></p> <ul style="list-style-type: none"> describe the basic features of OOP: <ul style="list-style-type: none"> classes, objects, attributes and behaviour, relationships member functions, class data and instantiation public and private members abstraction and data encapsulation use of constructor and operator overloading class libraries explain key differences between OOP and procedure-oriented programming in: <ul style="list-style-type: none"> designing programs the storage and access of data maintenance of programs give examples of how OOP facilitates: <ul style="list-style-type: none"> real-world modelling data integrity code reliability code maintainability code reusability. 	<p>Object-oriented programming is the current industry paradigm. It is more of an enhancement than a replacement of procedure-oriented programming.</p> <p>Good OOP relies on good procedure-oriented program practice to create well-designed objects, just as good procedure-oriented program practice relies on structured programming to create good procedures.</p>
Object-oriented Analysis, Design and Development	<ul style="list-style-type: none"> use iterative and incremental approaches in the analysis, design and development (architecture) stages of the software development process identify/describe, in general terms, the domain in which a specific program will be used; i.e., domain analysis develop general use cases to describe how users, and other systems, will apply the projected program combine the results of the domain analysis and the use cases to create a general design model and outline the architecture of the program identify/describe problems and/or problem components best solved using built-in class libraries—compiler provided libraries—and class libraries provided by other programmers identify/describe problems and/or problem components best solved using programmer built-in class libraries 	<p>Procedure-oriented software development was dominated by the Waterfall Methodology, in which software development was one-way and discrete. Object-oriented software development is iterative and incremental. Design will start once analysis has identified the basic nature of the problem, and development (architecture) will be started as soon as design has provided a general description.</p>

(continued)

COURSE INF2220: OBJECT-ORIENTED PROGRAMMING 1 (continued)

Concept	Specific Outcomes	Notes
(continued) Object-oriented Analysis, Design and Development	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • create simple class diagrams, describing the attributes and methods of the required classes, and interaction diagrams, showing the interaction among objects • work in a team with other students to carry out object-oriented design tasks. 	
Object-oriented Implementation, Testing and Maintenance	<ul style="list-style-type: none"> • use iterative and incremental approaches in the implementation, testing and maintenance phases of the software development process • modify/create objects from existing class libraries • modify/create classes and incorporate them in class libraries • create a prototype based on the design, using built-in class libraries—compiler provided libraries—class libraries provided by other programmers, or custom-created classes and objects • test and modify the prototype • repeat the coding/testing cycle; i.e., an iterative coding cycle, to add additional features to the prototype • profile and optimize the code for delivery • use overloaded constructors and operators where required to recast existing code and provide additional functionality to classes • work in a team with other students to carry out OOP tasks. 	<p>Procedure-oriented implementation, testing and maintenance were also dominated by the Waterfall Methodology, in which code construction was one-way and discrete. Object-oriented implementation and testing is iterative and incremental. Sometimes described as iterative prototyping, the process calls for the creation of a barebones executable prototype, which is tested and debugged before additional features are added. This cycle of implementing the design model and testing the resulting code is repeated until the design model is fully coded.</p>

COURSE INF2220: OBJECT-ORIENTED PROGRAMMING 1 (continued)

Concept	Specific Outcomes	Notes
Object-oriented Computer Language Syntax	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • use the structured programming approaches developed in INF1080 Programming 1 and INF2150 Programming 2, and the procedure-oriented approaches developed in INF2160 Programming 3 to: <ul style="list-style-type: none"> – modify/create objects from pre-existing classes – add member functions to existing classes – modify/create simple classes – use constructor and operator overloading to add power – organize these classes into appropriate class libraries. 	Structured programming is basic to good procedure-oriented programming, and both are basic to good object-oriented programming.
Workstation Management	<ul style="list-style-type: none"> • apply efficient workstation position and routines that encourage: <ul style="list-style-type: none"> – good health and safety—posture, positioning of hardware and furniture – security for hardware, software, supplies and personal work • demonstrate efficient and appropriate use of time and resources, including: <ul style="list-style-type: none"> – start-up procedures – organization of work area – closing procedures • apply effective decision-making strategies in programming assignments, including: <ul style="list-style-type: none"> – planning activities – organizing data, information, resources – considering alternatives – evaluating activities/results • use related terminology to describe basic processes, procedures and tools. 	

COURSE INF2210: COMPUTER SCIENCE 2**Level:** Intermediate**Theme:** Computer Science**Prerequisite:** INF1210 Computer Science 1

Description: Students extend their knowledge of the discipline of computer science by exploring the modular paradigm and its impact on algorithm development and implementation (programming). Students also add to their understanding by exploring the stylized von Neumann computer system at the machine level, and by examining the impact of computer science and computer technology on society.

Parameters: Designed to be taught in conjunction with INF2160 Programming 3, INF2170 Programming 4, INF2200 Information Highway 2 and INF2130 Multimedia Authoring 2 as a Grade 11 course in Computer Science.

Curriculum and Assessment Standards

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<i>The student will:</i> <ul style="list-style-type: none">identify and describe past, present and future trends in the development of computer technology	<i>Assessment of student achievement should be based on:</i> <ul style="list-style-type: none">a test, presentation or project designed to address the following topics:<ul style="list-style-type: none">the qualitative change/growth in computer applications from the recent past, through the present and into the near future. Note: Qualitative changes are changes such as the shift from computing in the recent past to communication in the present to bionics in the futurethe quantitative spread of computer technology through society from the recent past, through the present and into the near future Note: Quantitative changes are changes in the relative importance of the technology to different sectors of society	10
<ul style="list-style-type: none">explain and demonstrate the nature, development, structure, use of key algorithms associated with modular approaches and application of these idioms to create complex algorithms	<ul style="list-style-type: none">a presentation or project designed to demonstrate:<ul style="list-style-type: none">the basic nature of the modular paradigm and its impact on algorithm design and developmentthe ability to design, develop and explain algorithms organized into a modular formatproficient use of key basic algorithms (idioms) and the ability to use idioms to create other, complex algorithms	25

COURSE INF2210: COMPUTER SCIENCE 2 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> explain and demonstrate the rationale, use of subprograms, procedural abstraction and treatment of data in modular programs 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> a presentation or project designed to explain and demonstrate: <ul style="list-style-type: none"> the basic nature of the modular paradigm and its impact on algorithm implementation or programming the ability to implement modular algorithms using appropriately coupled subprograms and data transfer techniques that reflect procedural abstraction the ability to carry out individual module testing as well as integration testing of the program as a whole 	25
<ul style="list-style-type: none"> explain and demonstrate the rationale, representation and key uses of the fundamental derived data types 	<ul style="list-style-type: none"> a presentation or project demonstrating: <ul style="list-style-type: none"> the basic nature, representation and utility of derived data types, such as arrays, sets (enums) and records (strues) 	20
<ul style="list-style-type: none"> explain and analyze the nature, operation and basic architecture of the von Neumann computer system at the machine level 	<ul style="list-style-type: none"> a presentation or project demonstrating the von Neumann computer architecture at the machine level to: <ul style="list-style-type: none"> explain the nature and function of the main components of the CPU analyze how these components work together to process data explain the basic nature of machine language <p><i>Assessment Tool</i> <i>Assessment Checklist: Computer Science 2 Concepts (Introductory), INF2210-1</i> <i>Sample Assignment: Computer Science 2 Concepts (Introductory), INF2210-2</i></p> <p><i>Standard</i> <i>Rating of 2 for the Problem-solving Phase and a rating of 3 for the Implementation Phase</i></p>	20
<ul style="list-style-type: none"> demonstrate basic competencies. 	<ul style="list-style-type: none"> observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	Integrated throughout

COURSE INF2210: COMPUTER SCIENCE 2 (continued)

Concept	Specific Outcomes	Notes
Aspects of Computer Science	<p><i>The student should:</i></p> <ul style="list-style-type: none"> describe the qualitative shift or expansion of the application of computer technology over time. Specifically the student should be able to describe/explain/analyze trends, such as the shift in focus: <ul style="list-style-type: none"> in the recent past on <ul style="list-style-type: none"> traditional computation information warehousing (databases) automation and cybernetics to the present focus on <ul style="list-style-type: none"> communication artificial intelligence to an emerging focus in the near future on <ul style="list-style-type: none"> bionics and cyborganization artificial life describe the quantitative expansion of the application of computer technology over time. Specifically the student should be able to describe/explain/analyze trends, such as: <ul style="list-style-type: none"> the expansion in the recent past from <ul style="list-style-type: none"> the military the scientific community the government large and medium-sized institutions the present's expansion into <ul style="list-style-type: none"> small institutions the home both industrial and domestic machines personal information managers a projected expansion into <ul style="list-style-type: none"> personal expert systems implanted systems artificial life 	

COURSE INF2210: COMPUTER SCIENCE 2 (continued)

Concept	Specific Outcomes	Notes
Algorithmic Problem Solving	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • carry out the first step of the Systems Development Life Cycle (Analysis) using modular approaches, such as: <ul style="list-style-type: none"> – problem parsing and decomposition – identification of subtasks – data structuring – operation identification • carry out the second step of the Systems Development Life Cycle (Design) using modular approaches, such as: <ul style="list-style-type: none"> – top-down design – step-wise refinement – reusable code segments – scope considerations with an emphasis on avoiding global data – modular implementation <ul style="list-style-type: none"> • using appropriate coupling approaches • using appropriate levels of cohesion – data dictionaries, where required – both industrial and domestic machines – bottom-up coding, where appropriate • demonstrate iterative and incremental approaches in the analysis and design stages of the software development process • add additional tools to the algorithm development toolkit, such as: <ul style="list-style-type: none"> – HIPO charting – structure diagrams – Warnier/Orr diagrams 	

COURSE INF2210: COMPUTER SCIENCE 2 (continued)

Concept	Specific Outcomes	Notes
	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • demonstrate a number of core algorithms with derived data types, such as: <ul style="list-style-type: none"> – arrays – traversing – sorting <ul style="list-style-type: none"> • exchange • insertion • selection – searching – merging 	
Implementing the Algorithm (Software and Software Development)	<ul style="list-style-type: none"> • demonstrate the third step of the Systems Development Life Cycle (Development or Coding) using modular approaches: <ul style="list-style-type: none"> – use of subprograms – procedures/functions – stub programming – prototyping – libraries • demonstrate derived data types, such as: <ul style="list-style-type: none"> – arrays – vectors – matrices – sets (enums) – records (structs) • demonstrate data representation, such as: <ul style="list-style-type: none"> – ASCII coding • demonstrate an ability to use iterative and incremental approaches in the Implementation and Maintenance stages of the Systems Development Life Cycle 	

COURSE INF2210: COMPUTER SCIENCE 2 (continued)

Concept	Specific Outcomes	Notes
Platform for Executing the Algorithm (Computer Systems)	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • demonstrate the machine level of a hypothetical von Neumann machine by describing/explaining/using: <ul style="list-style-type: none"> – the basic components of the CPU <ul style="list-style-type: none"> • ALU • control unit • registers • program counter • instruction register – the bus – the memory • demonstrate the machine language of a hypothetical von Neumann machine by describing/explaining/using: <ul style="list-style-type: none"> – op codes – operands – symbolic representation • demonstrate the machine-level operations of a hypothetical von Neumann machine by describing/explaining/using: <ul style="list-style-type: none"> – the machine cycle <ul style="list-style-type: none"> • fetch • decode • execute – the flow of data through the computer under the direction of a machine-language program • demonstrate the mediating role played by system software between the human level and machine level: <ul style="list-style-type: none"> – operating systems – language translators – memory managers – information managers – scheduler – utilities. 	

COURSE CURRICULUM AND ASSESSMENT STANDARDS:

SECTION F: ADVANCED LEVEL

The following pages define the curriculum and assessment standards for the advanced level of Information Processing.

Advanced level courses demand a higher level of expertise and help prepare students for entry into the workplace or a related post-secondary program.

Course INF3010:	Hardware/Software Analysis	F.3
Course INF3020:	Local Area Networks	F.7
Course INF3030:	Keyboarding 4	F.13
Course INF3040:	Keyboarding 5	F.17
Course INF3050:	Keyboarding 6	F.21
Course INF3060:	Word Processing 3	F.25
Course INF3070:	Electronic Publishing 2	F.31
Course INF3080:	Information Management Tools	F.35
Course INF3090:	Software Integration 3	F.39
Course INF3100:	Specialization 1	F.43
Course INF3110:	Specialization 2	F.47
Course INF3120:	Software Integration 2	F.51
Course INF3130:	Multimedia Authoring 2	F.55
Course INF3140:	Expert Systems	F.59
Course INF3150:	Programming Application 1	F.63
Course INF3160:	Programming Application 2	F.67
Course INF3170:	Programming Application 3	F.71
Course INF3180:	Telecommunications 2	F.75
Course INF3190:	Information Highway 3	F.79
Course INF3200:	Internet Services	F.83
Course INF3220:	Object-oriented Programming 2	F.87
Course INF3230:	Dynamic Data Structures 1	F.93
Course INF3240:	Dynamic Data Structures 2	F.97
Course INF3210:	Computer Science 3	F.101

COURSE INF3030: KEYBOARDING 4**Level:** Advanced**Theme:** Text/Data Input**Prerequisite:** None**Description:** Students develop their text and data keyboarding skills to entry-level occupational expectations.**Parameters:** Computer workstation, disk, word processing software, support resources.**Curriculum and Assessment Standards**

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> demonstrate proficient keyboarding competence: <ul style="list-style-type: none"> text entry at 50 words per minute (wpm) numeric entry at 150 keystrokes per minute (kpm) technique 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> three timed writings, each from different straight copy material, over a period of no more than five consecutive class periods, which demonstrates proper touch keyboarding : <ul style="list-style-type: none"> on alphabetic keys <ul style="list-style-type: none"> three-minute duration maximum one uncorrected error SI 1.3 – 1.4 50 words per minute on numeric keys: <ul style="list-style-type: none"> one-minute duration maximum one uncorrected error 150 numeric keystrokes a minute on 1 to 5 digit numbers observations over the last quarter of the learning period, during timings and drill work. <p><i>Assessment Tool</i> <i>Assessment Checklist: Text-Data Entry (INFTDENT)</i></p> <p><i>Standard</i> <i>Rating of:</i> 3 – Eye Focus 3 – Keystroking 3 – Service Keys 3 – Body Position</p> 	<p>50</p> <p>10</p> <p>30</p>

COURSE INF3030: KEYBOARDING 4 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • apply, consistently, appropriate workstation routines • demonstrate basic competencies. 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • demonstrate appropriate workstation routines. <p><i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i></p> <p><i>Standard</i> <i>Rating of:</i> 3 – Workstation Use 3 – File Management 3 – Time Management/Organization 3 – Professionalism</p> <ul style="list-style-type: none"> • observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	<p>10</p> <p>Integrated throughout</p>

Concept	Specific Outcomes	Notes
Text Entry	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • demonstrate increasingly rapid, accurate touch keystroking on straight and draft (edited) copy of: <ul style="list-style-type: none"> – alphanumeric keys – all punctuation keys – service keys (enter, shift, delete, backspace, tab) • use function and cursor movement key efficiently • demonstrate correct keystroking technique: <ul style="list-style-type: none"> – enter text using designated fingers – maintain home-row anchor position – demonstrate correct posture (hands, arms, body) 	<p>Develop speed and accuracy at the phrase, sentence and short paragraph level using short, repetitive timings (.5 to one minute) with straight copy text of varying SI. (1.2–1.6).</p> <p>Draft copy should include basic spacing, spelling, punctuation and spacing errors (no more than one error per every 10 words).</p>

COURSE INF3040:	KEYBOARDING 5
Level:	Advanced
Theme:	Text/Data Input
Prerequisite:	None
Description:	Students increase their occupational-level keyboarding competence of text, data and function/service keys, using straight copy and edited material.

Theme: Text/Data Input

Description: Students increase their occupational-level keyboarding competence of text, data and function/service keys, using straight copy and edited material.

Curriculum and Assessment Standards

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • demonstrate proficient keyboarding competence: <ul style="list-style-type: none"> – text entry at 60 words per minute (wpm) – numeric entry at 180 keystrokes per minute (kpm) – technique 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • three timed writings, each from different straight copy material, over a period of no more than five consecutive class periods, which demonstrates proper touch keyboarding: <ul style="list-style-type: none"> – on alphabetic keys: <ul style="list-style-type: none"> • three-minute duration • maximum one uncorrected error • SI ≥ 1.35 • 60 words per minute – on numeric keys: <ul style="list-style-type: none"> • one-minute duration • maximum one uncorrected error • 180 numeric keystrokes a minute on 1 to 6 digit numbers – observations over the last quarter of the learning period, during timings and drill work. <p><i>Assessment Tool</i> <i>Assessment Checklist: Text–Data Entry (INFTDENT)</i></p> <p><i>Standard</i> <i>Rating of:</i> 3 – Eye Focus 3 – Keystroking 3 – Service Keys 3 – Body Position</p> 	<p>50</p> <p>20</p> <p>20</p>

COURSE INF3040: KEYBOARDING 5 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • apply, consistently, appropriate workstation routines • demonstrate basic competencies. 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • demonstrate appropriate workstation routines. <p><i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i></p> <p><i>Standard</i> <i>Rating of:</i> 3 – Workstation Use 3 – File Management 3 – Time Management/Organization 3 – Professionalism</p> <ul style="list-style-type: none"> • observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	<p>10</p> <p>Integrated throughout</p>

Concept	Specific Outcomes	Notes
Text Entry	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • demonstrate increasingly rapid, accurate touch keystroking on straight and draft copy (edited) of: <ul style="list-style-type: none"> – alphanumeric keys – all punctuation keys – service keys • use function and cursor movement keys efficiently • demonstrate correct keystroking technique: <ul style="list-style-type: none"> – enter text using designated fingers – maintain home-row anchor position – demonstrate correct posture (hands, arms, body) • proofread and edit text (screen and hard copy) to ensure text is without error 	<p>Enter, shift, delete, backspace, tab.</p> <p>Develop speed and accuracy at the phrase, sentence and short paragraph level using short, repetitive timings (.5 to one minute) with straight copy text of varying SI (1.2–1.6).</p> <p>Draft copy should include basic spacing, spelling, punctuation and spacing errors (no more than one error per every 10 words).</p>

COURSE INF3050:	KEYBOARDING 6
Level:	Advanced
Theme:	Text/Data Input
Prerequisite:	None
Description:	Students enhance their occupational-level keyboarding competence of all keystroke functions, using unedited, edited and straight copy material.
Parameters:	Computer workstation, disk, word processing software, support resources.

Description: Students enhance their occupational-level keyboarding competence of all keystroke functions, using unedited, edited and straight copy material.

Parameters: Computer workstation, disk, word processing software, support resources.

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • demonstrate proficient keyboarding competence: <ul style="list-style-type: none"> – text entry at 70 words per minute (wpm) – numeric entry at 200 keystrokes per minute (kpm) – technique 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • three timed writings each from different straight copy material, over a period of no more than five consecutive class periods, which demonstrates proper touch keyboarding : <ul style="list-style-type: none"> – on alphabetic keys: <ul style="list-style-type: none"> • three-minute duration • maximum one uncorrected error • SI ≥ 1.35 • 70 words per minute – on numeric keys: <ul style="list-style-type: none"> • one-minute duration • maximum one uncorrected error • 200 numeric keystrokes a minute on 1 to 6 digit numbers – observations over the last quarter of the learning period, during timings and drill work. <p><i>Assessment Tool</i> <i>Assessment Checklist: Text–Data Entry (INFTDENT)</i></p> <p><i>Standard</i> <i>Rating of:</i> 3 – Eye Focus 3 – Keystroking 3 – Service Keys 3 – Body Position</p> 	<p>50</p> <p>20</p> <p>20</p>

COURSE INF3050: KEYBOARDING 6 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • apply, consistently, appropriate workstation routines • demonstrate basic competencies. 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • demonstrate appropriate workstation routines. <p><i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i></p> <p><i>Standard</i> <i>Rating of:</i> 3 – Workstation Use 3 – File Management 3 – Time Management/Organization 3 – Professionalism</p> <ul style="list-style-type: none"> • observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	<p>10</p> <p>Integrated throughout</p>

Concept	Specific Outcomes	Notes
Text/Data Entry	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • use formatted, straight-copy material as well as unformatted rough-draft material • touch-keystroke alphabetic, numeric, punctuation, service keys • consistently apply: <ul style="list-style-type: none"> – correct finger/key placement – healthful body position – acceptable eye/copy focus • use numeric keys and/or number pad. 	<p>A few five-minute timed attempts can be used to prepare for workplace expectations if deemed appropriate.</p>

COURSE INF3220: OBJECT-ORIENTED PROGRAMMING 2 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> – analyzing/revising/creating object-oriented programs that use templated, base and derived classes 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> – determine if any of the objects could be instantiated using a templated class or classes – determine if any of the objects form a class hierarchy that could employ composition or inheritance – identify the relationship among objects, with a focus on identifying data access requirements – create a diagram describing the objects and their relationships that can be used to guide program construction. <p>the programs should demonstrate the ability to code object-oriented programs that use:</p> <ul style="list-style-type: none"> – destructors – constructor and operator overloading – friend functions – templated classes – simple composition and inheritance – simple class hierarchies <p><i>Assessment Tool</i> <i>Assessment Checklist: Object-oriented Programming 2, INF3220-1</i> <i>Sample Assignment: Object-oriented Programming 2, INF3220-2</i></p> <p><i>Standard</i> <i>Rating of 2 for the Problem-solving Phase and a rating of 3 for the Implementation Phase</i></p>	45%
<ul style="list-style-type: none"> • apply, consistently, appropriate workstation routines 	<ul style="list-style-type: none"> • demonstrations of appropriate workstation routines <p><i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i></p> <p><i>Standard</i> <i>Rating of:</i> 2 – Workstation Routines 3 – File Management 2 – Time Management/Organization 3 – Professionalism</p>	10%
<ul style="list-style-type: none"> • demonstrate basic competencies. 	<ul style="list-style-type: none"> • observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	Integrated throughout

COURSE INF3220: OBJECT-ORIENTED PROGRAMMING 2 (continued)

Concept	Specific Outcomes	Notes
Nature of Object-oriented Programming (OOP)	<p><i>The student should:</i></p> <ul style="list-style-type: none"> describe features of OOP, including: <ul style="list-style-type: none"> use of private, public and protected members, accessors and modifiers to control access to data use of templated classes with parameterized data use of constructor and operator overloading, composition, polymorphism and inheritance to recast existing code to improve production efficiencies use of objects; client/server relationship between classes; and templated, base and derived classes to more closely model real-world situations explain key differences between OOP and procedure-oriented programming in: <ul style="list-style-type: none"> designing programs the storage and access of data maintenance of programs give examples of how OOP facilitates: <ul style="list-style-type: none"> real-world modelling data integrity code reliability code maintainability code reusability. 	<p>Three reasons why object-oriented programming has come to prominence are improved data security, improved production cycle and an improved ability to write code that more closely corresponds to the real world.</p> <p>Improved data security is primarily delivered through controlled access to data. Improved production cycle is delivered through programming efficiencies resulting from the ability to reuse code. An improved ability to write code is based on the correspondence of the object construct to the real world.</p>
Object-oriented Analysis, Design and Development (OOD) (continued)	<ul style="list-style-type: none"> continue to use iterative and incremental approaches in the analysis, design and development (architecture) stages of the software development process identify/describe in some detail the domain in which a specific program will be used; i.e., domain analysis develop and structure a set of use cases that give a detailed description of how users and other systems will apply the projected program combine the results of the domain analysis and the use cases to create a general design model; and outline the architecture of the program, using any appropriate notation; e.g., structured diagram, input/processing/output (IPO) chart, Warnier-Orr diagram 	<p>Object-oriented design (OOD) is an evolving process that is still not fully developed. Until recently, analysis and design was dominated by approaches developed when procedure-oriented programming was paramount. In recent years, procedure-oriented approaches—flow charts, hierarchical charts, IPO charts, Warnier-Orr diagrams—are being replaced by OOD specific approaches: e.g., object charts, class diagrams.</p>

COURSE INF3220: OBJECT-ORIENTED PROGRAMMING 2 (continued)

Concept	Specific Outcomes	Notes
<p>(continued)</p> <p>Object-oriented Analysis, Design and Development (OOD)</p>	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • identify/describe problems and/or problem components best solved using built-in class libraries—compiler provided libraries—or class libraries provided by other programmers • identify/describe problems and/or problem components best solved using programmer built-in class libraries • identify/describe problems and/or problem components best solved using templated classes • identify/describe problems and/or problem components best solved using composition • identify/describe problems and/or problem components best solved using base and derived classes • create simple class diagrams, describing the attributes and methods of the required classes, and interaction diagrams, showing the interaction among objects • design/define the required member functions, constructors and operators • work in a team with other students to carry out OOD tasks. 	<p>At this point in their learning, students should develop a general approach to analysis and design that combines the modularity associated with procedure-oriented programming and the iterative and incremental processes associated with OOD. A formal understanding of OOD notation, such as that found in the emerging unified modeling language (UML) approach, is not required at this time.</p>
<p>Object-oriented Implementation, Testing and Delivery</p> <p>(continued)</p>	<ul style="list-style-type: none"> • continue to use iterative and incremental approaches in the implementation, testing and delivery of the software development process • modify/create objects, using existing class libraries • modify/create classes and incorporate them in class libraries • modify/create templated classes • use class containment, where appropriate • modify/create base and derived classes • use overloaded constructors and operators where required to recast existing code and provide additional functionality to classes • establish class relationships 	

COURSE INF3220: OBJECT-ORIENTED PROGRAMMING 2 (continued)

Concept	Specific Outcomes	Notes
(continued) Object-oriented Implementation, Testing and Delivery	<i>The student should:</i> <ul style="list-style-type: none"> • create a prototype based on the design, using built-in class libraries—compiler provided libraries—class libraries provided by other programmers, or custom-created classes and objects • test and modify the prototype • repeat the coding/testing cycle; i.e., an iterative coding cycle, to add more features to the prototype • profile and optimize the code for delivery • work in a team with other students to carry out OOD tasks. 	
Object-oriented Computer Language Syntax	<ul style="list-style-type: none"> • apply the structured programming approaches developed in INF1080 Programming 1 and INF2150 Programming 2, the procedure-oriented approaches developed in INF2160 Programming 3 and the introduction to objects provided in Object-oriented Programming 1 to: <ul style="list-style-type: none"> – use accessors and modifiers to control class access – use constructor and operator overloading to add power – use templated classes, composition, inheritance and polymorphism to promote code reusability. 	Structured programming is a prerequisite to good procedure-oriented programming, just as good procedure-oriented programming is a prerequisite to good object-based programming. Object-based programming must be understood before object-oriented programming can be mastered.
Workstation Management (continued)	<ul style="list-style-type: none"> • apply efficient workstation position and routines that encourage: <ul style="list-style-type: none"> – good health and safety—posture, positioning of hardware and furniture – security for hardware, software, supplies and personal work • demonstrate efficient and appropriate use of time and resources, including: <ul style="list-style-type: none"> – start-up procedures – organization of work area – closing procedures 	

COURSE INF3220: OBJECT-ORIENTED PROGRAMMING 2 (continued)

Concept	Specific Outcomes	Notes
(continued) Workstation Management	<i>The student should:</i> <ul style="list-style-type: none">• apply effective decision-making strategies in programming assignments, including:<ul style="list-style-type: none">– planning activities– organizing data, information, resources– considering alternatives– evaluating activities/results• use related terminology to describe basic processes, procedures and tools.	

COURSE INF3230: DYNAMIC DATA STRUCTURES 1**Level:** Advanced**Theme:** Programming**Prerequisite:** INF2180 Programming 5**Description:** Students are formally introduced to dynamic data structures in general and to linked lists in particular.**Parameters:** Access to appropriate computer equipment and software. Specifically, students must have access to an object-oriented programming (OOP) environment that encourages a formal treatment of objects.**Supporting Courses:** INF3150 Programming Application 1
INF3220 Object-oriented Programming 2**Curriculum and Assessment Standards**

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<i>The student will:</i> <ul style="list-style-type: none">• identify/describe the concepts of dynamic memory allocation and basic dynamic data structures, and explain their advantages over more static structures	<i>Assessment of student achievement should be based on:</i> <ul style="list-style-type: none">• a teacher-directed evaluation designed to test the student's ability to describe and illustrate:<ul style="list-style-type: none">– abstract data types (ADTs)– the structure of computer memory and the concept of free store (the heap)– pointers, nodes and related concepts– the advantages of dynamic data structures, such as linked lists, over more static structures	20%
<ul style="list-style-type: none">• revise/create algorithms that make effective use of dynamic data structures to solve problems	<ul style="list-style-type: none">• a series of student-developed programs that demonstrate the efficient use of algorithms and language syntax <p>the algorithms should demonstrate the ability to:</p> <ul style="list-style-type: none">– identify problems that are best solved using linked lists– identify the input, processing and output commands needed to use linked lists– create an appropriate diagram/description of the proposed program	20%

COURSE INF3230: DYNAMIC DATA STRUCTURES 1 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> revise/create programs that make effective use of linked lists 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> a series of student developed programs that demonstrate the efficient use of algorithms and language syntax <p>the programs should demonstrate the ability to make appropriate use of:</p> <ul style="list-style-type: none"> pointers linked lists <p><i>Assessment Tool</i> <i>Assessment Checklist: Dynamic Data Structures 1, INF3230-1</i> <i>Sample Assignment: Dynamic Data Structures 1, INF3230-2</i></p> <p><i>Standard</i> <i>Rating of 2 for the Problem-solving Phase and a rating of 3 for the Implementation Phase</i></p>	50%
<ul style="list-style-type: none"> apply, consistently, appropriate workstation routines 	<ul style="list-style-type: none"> demonstrations of appropriate workstation routines <p><i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i></p> <p><i>Standard</i> <i>Rating of:</i> 2 – Workstation Routines 3 – File Management 2 – Time Management/Organization 3 – Professionalism</p>	10%
<ul style="list-style-type: none"> demonstrate basic competencies. 	<ul style="list-style-type: none"> observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	Integrated throughout

COURSE INF3230: DYNAMIC DATA STRUCTURES 1 (continued)

Concept	Specific Outcomes	Notes
Nature of Linked Lists	<p><i>The student should:</i></p> <ul style="list-style-type: none"> describe/illustrate the general nature of dynamic data structures, including: <ul style="list-style-type: none"> the mechanics of dynamic memory allocation: <ul style="list-style-type: none"> the structure of computer memory the heap—allocating and de-allocating memory pointers and related concepts linear and nonlinear data structures the advantages and disadvantages of dynamic data structures in relation to static data structures describe/illustrate the nature and mechanics of linked lists: <ul style="list-style-type: none"> the linked list as an abstract data type the logical structure of the linked list nodes, data elements, pointers, head pointers and external pointers. 	<p>Static data structures, such as ordinary arrays, occupy the same amount of memory every time the program is run. Dynamic data structures are used when data storage requirements are not known at the time the program is coded or when the storage requirements will change over the course of the program.</p> <p>An abstract data type is an abstract description of a data structure with the emphasis on its properties, functionality and use rather than on its specific implementation. It defines an interface or set of access methods to a collection of organized data without providing specifics as to how that data is organized or accessed.</p>
Analysis and Design Using Linked Lists	<ul style="list-style-type: none"> analyze/modify/create design documents that demonstrate the ability to: <ul style="list-style-type: none"> make appropriate use of dynamic program elements and data structures, such as pointers and linked lists identify the initialization, input, processing, output and termination requirements needed to use linked lists create an appropriate diagram/description of the proposed program. 	
Coding Using Linked Lists (continued)	<ul style="list-style-type: none"> analyze/modify/create programs that demonstrate the ability to make appropriate use of: <ul style="list-style-type: none"> dynamic memory <ul style="list-style-type: none"> memory allocation operators memory de-allocation operators pointers <ul style="list-style-type: none"> de-referencing operators address-of operators structure pointer operators 	

COURSE INF3230: DYNAMIC DATA STRUCTURES 1 (continued)

Concept	Specific Outcomes	Notes
(continued) Coding Using Linked Lists	<p><i>The student should:</i></p> <ul style="list-style-type: none"> – linked lists <ul style="list-style-type: none"> • node declaration • pointer declaration • link list initialization • appending nodes • accessing nodes • inserting nodes • deleting individual nodes • displaying the list • disposing of the list • doubly linked list (optional) • circular linked list (optional). 	
Workstation Management	<ul style="list-style-type: none"> • apply efficient workstation position and routines that encourage: <ul style="list-style-type: none"> – good health and safety—posture, positioning of hardware and furniture – security for hardware, software, supplies and personal work • demonstrate efficient and appropriate use of time and resources, including: <ul style="list-style-type: none"> – start-up procedures – organization of work area – closing procedures • apply effective decision-making strategies in programming assignments, including: <ul style="list-style-type: none"> – planning activities – organizing data, information, resources – considering alternatives – evaluating activities/results • use related terminology to describe basic processes, procedures and tools. 	

COURSE INF3240: DYNAMIC DATA STRUCTURES 2**Level:** Advanced**Theme:** Programming**Prerequisite:** INF3230 Dynamic Data Structures 1**Description:** Students add to their understanding of dynamic data structures by developing introductory algorithms and programs that use stacks, queues and trees.**Parameters:** Access to appropriate computer equipment and software. Specifically, students must have access to an object-oriented programming (OOP) environment that encourages a formal treatment of objects.**Supporting Courses:** INF3150 Programming Application 1
INF2180 Programming 5
INF3220 Object-oriented Programming 2**Curriculum and Assessment Standards**

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<i>The student will:</i> <ul style="list-style-type: none">identify/describe the stack, queue and tree Abstract Data Types	<i>Assessment of student achievement should be based on:</i> <ul style="list-style-type: none">a teacher-directed evaluation designed to test the student's ability to describe and illustrate:<ul style="list-style-type: none">stacksqueuestreesthe advantages and disadvantages of each Abstract Data Type (ADT) as a data structure	20%
<ul style="list-style-type: none">revise/create algorithms that make effective use of stacks, queues and trees to solve problems	<ul style="list-style-type: none">a series of student-developed programs that demonstrate the efficient use of algorithms and language syntax <p>the algorithms should demonstrate the ability to:</p> <ul style="list-style-type: none">identify problems that are best solved using stacks, queues and treesidentify the appropriate data structure to be used in a particular problemidentify the input, processing and output commands needed to use the data structurecreate an appropriate diagram/description of the proposed program	20%

COURSE INF3240: DYNAMIC DATA STRUCTURES 2 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> revise/create programs that make effective use of stacks, queues and trees 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> a series of student developed programs that demonstrate the efficient use of algorithms and language syntax <p>the programs should demonstrate the ability to make appropriate use of:</p> <ul style="list-style-type: none"> stacks queues trees <p><i>Assessment Tool</i> <i>Assessment Checklist: Dynamic Data Structures 2, INF3240-1</i> <i>Sample Assignment: Dynamic Data Structures 2, INF3240-2</i></p> <p><i>Standard</i> <i>Rating of 2 for the Problem-solving Phase and a rating of 3 for the Implementation Phase</i></p>	50%
<ul style="list-style-type: none"> apply, consistently, appropriate workstation routines 	<ul style="list-style-type: none"> demonstrations of appropriate workstation routines <p><i>Assessment Tool</i> <i>Assessment Checklist: Workstation Routines and Management (INFWRKSTN)</i></p> <p><i>Standard</i> <i>Rating of:</i> 2 – Workstation Routines 3 – File Management 2 – Time Management/Organization 3 – Professionalism</p>	10%
<ul style="list-style-type: none"> demonstrate basic competencies. 	<ul style="list-style-type: none"> observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	Integrated throughout

COURSE INF3240: DYNAMIC DATA STRUCTURES 2 (continued)

Concept	Specific Outcomes	Notes
Nature of Stacks, Queues and Trees	<p><i>The student should:</i></p> <ul style="list-style-type: none"> describe/illustrate the general nature of stacks, queues and trees, including: <ul style="list-style-type: none"> the general approach to data storage and manipulation in each structure the advantages and disadvantages of each data structure describe/illustrate the nature and mechanics of: <ul style="list-style-type: none"> stacks—last in first out (LIFO) queues—first in first out (FIFO) trees—roots, nodes, branch nodes, leaf node or terminal nodes, siblings, parent and child relationships. 	
Analysis and Design Using Stacks, Queues and Trees	<ul style="list-style-type: none"> analyze/modify/create design documents that demonstrate the ability to: <ul style="list-style-type: none"> make appropriate use of dynamic program elements and data structures, including: <ul style="list-style-type: none"> stacks queues trees identify the initialization, input, processing, output and termination requirements needed to use the data structure create an appropriate diagram/description of the proposed program. 	
Coding Using Stacks, Queues and Trees	<ul style="list-style-type: none"> analyze/modify/create programs that demonstrate the ability to make appropriate use of: <ul style="list-style-type: none"> stacks <ul style="list-style-type: none"> implementing a stack, using an array implementing a stack, using a linked list implementing a stack, using a class pushing and popping a stack—LIFO queues <ul style="list-style-type: none"> implementing a queue, using a linked list implementing a queue, using a class enqueue and dequeue operations—FIFO trees (binary) <ul style="list-style-type: none"> implementing trees, using a class appending nodes accessing nodes inserting nodes deleting nodes traversing trees binary search. 	

COURSE INF3240: DYNAMIC DATA STRUCTURES 2 (continued)

Concept	Specific Outcomes	Notes
Workstation Management	<p><i>The student should:</i></p> <ul style="list-style-type: none">• apply efficient workstation position and routines that encourage:<ul style="list-style-type: none">– good health and safety—posture, positioning of hardware and furniture– security for hardware, software, supplies and personal work• demonstrate efficient and appropriate use of time and resources, including:<ul style="list-style-type: none">– start-up procedures– organization of work area– closing procedures• apply effective decision-making strategies in programming assignments, including:<ul style="list-style-type: none">– planning activities– organizing data, information, resources– considering alternatives– evaluating activities/results• use related terminology to describe basic processes, procedures and tools.	

COURSE INF3210: COMPUTER SCIENCE 3 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> • explain and demonstrate the rationale, use of classes and objects, encapsulation and procedural abstraction, and treatment of data in object-oriented programs • explain the rationale and use of recursion and introductory recursive operations • identify and demonstrate the rationale, creation and manipulation of files and Abstract Data Types • explain and analyze the nature, operation, basic architecture and utility of a Turing machine 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> • a presentation or project designed to explain and demonstrate: <ul style="list-style-type: none"> – the basic nature of the object-oriented paradigm and its impact on algorithm implementation or programming – the ability to implement object-oriented algorithms, such as object diagrams using classes and appropriate data-handling techniques – the ability to carry out the testing of individual classes and objects as well as integration testing of the program as a whole • a presentation or project designed to explain: <ul style="list-style-type: none"> – rationale for – use of, and – advantages and disadvantages of recursion and recursive processes • a presentation or project designed to explain and demonstrate: <ul style="list-style-type: none"> – an understanding of the basic nature, creation and utility of different types of files – the basic nature, creation and utility of Abstract Data Types – an ability to use simpler data structures to create Abstract Data Structures • a presentation or project explaining and analyzing the nature of a Turing machine in terms of: <ul style="list-style-type: none"> – a model of a computing agent – the basic configuration – programming – the Church-Turing thesis – unsolvable problems <p><i>Assessment Tool</i> <i>Assessment Checklist: Computer Science 3 Concepts (Senior), INF3210-1</i> <i>Sample Assignment: Computer Science 3 Concepts (Senior), INF3210-2</i></p> <p><i>Standard</i> <i>Rating of 2 for the Problem-solving Phase and a rating of 3 for the Implementation Phase</i></p>	<p>20</p> <p>10</p> <p>20</p> <p>20</p>

COURSE INF3210: COMPUTER SCIENCE 3 (continued)

General Outcomes	Assessment Criteria and Conditions	Suggested Emphasis
<p><i>The student will:</i></p> <ul style="list-style-type: none"> demonstrate basic competencies. 	<p><i>Assessment of student achievement should be based on:</i></p> <ul style="list-style-type: none"> observations of individual effort and interpersonal interaction during the learning process. <p><i>Assessment Tool</i> <i>Basic Competencies Reference Guide and any assessment tools noted above</i></p>	<p>Integrated throughout</p>

Concept	Specific Outcomes	Notes
<p>Aspects of Computer Science</p> <p>(continued)</p>	<p><i>The student should:</i></p> <ul style="list-style-type: none"> demonstrate the broad sequence of events variously described as the post industrial, the cybernetic and the information revolution. Specifically the student should be able to describe/explain/utilize: <ul style="list-style-type: none"> the equivalency of this revolution with other similar revolutions, such as the Neolithic-Urban Revolution and the Industrial Revolution, with a focus on <ul style="list-style-type: none"> the initiating effect of a technological paradigm shift (i.e., Industrial: shift from animal-powered, human-directed tools to machine-powered, human-directed tools (mechanization); Information: shift from human-directed, machine-powered tools to machine-directed, machine-powered tools—automation) the ripple effect of the paradigm shift through society plotting further change as our societies continue to move through the revolution the general economic impact <ul style="list-style-type: none"> eclipse of industrial economies emergence of information economy the service economy globalized economies the social impact <ul style="list-style-type: none"> new class structure pace of life (24-7) mobility demographic shifts 	

COURSE INF3210: COMPUTER SCIENCE 3 (continued)

Concept	Specific Outcomes	Notes
(continued) Aspects of Computer Science	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • fragmented families • privacy issues – the political impact <ul style="list-style-type: none"> • decline of the nation–state • transnational corporations • supra-national constellations • local autonomy • remote control war • changes in political ideologies: new forms of democracy and authoritarianism 	
Algorithmic Problem Solving	<ul style="list-style-type: none"> • carry out the first step of the Systems Development Life Cycle (Analysis) using object-oriented concepts, such as: <ul style="list-style-type: none"> – encapsulation, inheritance and polymorphism – classes, objects, instantiation – member and data functions – public and private access modifiers – class libraries • design programs that are: <ul style="list-style-type: none"> – correct, reliable, efficient, robust – extendable, scalable – reusable, portable – maintainable • use object-oriented design techniques, such as: <ul style="list-style-type: none"> – requirement analysis <ul style="list-style-type: none"> • use cases • domain analysis • object diagrams – iterative class design <ul style="list-style-type: none"> • principal classes • elaborate object diagrams • CRC cards • prototype • employ appropriate relationships: <ul style="list-style-type: none"> – dependency – association – aggregation – composition 	
(continued)		

COURSE INF3210: COMPUTER SCIENCE 3 (continued)

Concept	Specific Outcomes	Notes
(continued) Algorithmic Problem Solving	<p><i>The student should:</i></p> <ul style="list-style-type: none"> • add additional tools to the algorithm development toolkit, such as: <ul style="list-style-type: none"> – object diagrams <ul style="list-style-type: none"> • class • activity • iteration diagrams • demonstrate a number of core recursive idioms, such as: <ul style="list-style-type: none"> – accumulation (keeping a running total) – calculating factorials – determining minimums and maximums – searches – sorts 	
Implementing the Algorithm (Software and Software Development)	<ul style="list-style-type: none"> • demonstrate the third step of the Systems Development Life Cycle (Development or Coding) using object-oriented approaches: <ul style="list-style-type: none"> – use of classes and objects – implementation of the user interface – prototyping – elaborate the hierarchy – develop libraries • demonstrate the use of files: <ul style="list-style-type: none"> – fields and records – sequential files – random access files • demonstrate abstract data types, using: <ul style="list-style-type: none"> – arrays – structs or records – classes 	
Platform for Executing the Algorithm (Computer Systems)	<ul style="list-style-type: none"> • demonstrate the principals of a Turing machine with an emphasis on being able to: <ul style="list-style-type: none"> – build/assemble a working Turing machine – demonstrate the ability to execute simple programs on the machine, such as a unary addition machine – interpret/design/write state diagrams – describe its nature as an ultimate model of a computing agent – describe/explain/utilize the Church-Turing thesis – describe the nature of unsolvable problems 	

TABLE OF CONTENTS

ASSESSING STUDENT ACHIEVEMENT

Assessing Student Achievement in CTS	G.5
Assessing Student Achievement in Information Processing	G.7

Assessment Tools Generic to CTS

Basic Competencies Reference Guide	G.8
Generic Rating Scale	G.10
Frameworks for Assessment	
CTSISS: Issue Analysis	G.11
CTSLAB: Lab Investigations	G.12
CTSNEG: Negotiation and Debate	G.13
CTSPRE: Presentations/Reports	G.14
CTSRES: Research Process	G.15

Assessment Tools Generic to Information Processing Strand

INF CRT: Assessment Checklist: Correspondence, Reports, Tables	G.16
INF DB: Assessment Checklist: Databases	G.17
INF EPDOC: Assessment Checklist: Electronic Publishing Document Production	G.18
INF EPSF: Assessment Checklist: Electronic Publishing Software Functions ..	G.19
INF INTEG: Assessment Checklist: Software Integration 1, 2 and 3	G.20
INF KEYNB: Reference Chart: Keyboarding and Numberpad Rates	G.21
INF MMDOC: Assessment Checklist: Multimedia Productions and Presentations	G.22
INF MMSF: Assessment Checklist: Multimedia Software Functions	G.23
INF PRGM1: Assessment Checklist: Introductory and Intermediate Programming	G.24
INF PRGM2: Assessment Checklist: Intermediate Programming	G.25
INF PRGM3: Assessment Checklist: Advanced Programming Applications	G.26
INF PSAM1: Programming: Sample Assignments 1A–3A	G.27
INF PSAM2: Programming: Sample Assignments 4A/B–5A/B	G.29
INF PSAM3: Programming: Sample Assignments PA1, 2, 3	G.31
INF SPEC: Assessment Checklist: Specialization 1 and 2	G.32
INF SS: Assessment Checklist: Spreadsheets	G.33
INF TDENT: Assessment Checklist: Text–Data Entry	G.34
INF WP: Assessment Checklist: Word Processing	G.35
INF WRKSTN: Assessment Checklist: Workstation Routines and Management	G.36

Assessment Tools Specific to Courses in the Information Processing Strand

INF1010–1: Assessment Checklist: A. File Management Procedures B. Text/Data Entry C. Computer Workstation Components ...	G.37
INF1010–2: Assessment Guide: Presentations and Reports	G.38
INF1090–1: Assessment Guide: Information Highway 1	G.39
INF2010–1: Assessment Guide: Workstation Operations	G.40

INF2140-1:	Assessment Guide: Process Control Project	G.41
INF2140-2:	Process Control Sample Project	G.42
INF2190-1:	Assessment Checklist: Telecommunication Systems Use	G.43
INF2190-2:	Assessment Checklist: Telecommunication Systems Presentation/Report	G.44
INF2190-3:	Assessment Checklist: Telecommunication Systems Infrastructure Presentation/Report	G.46
INF2200-1:	Assessment Guide: Information Highway 2	G.47
INF3010-1:	Presentations/Reports: Hardware/Software Analysis	G.48
INF3020-1:	Assessment Guide: Local Area Networks Project	G.49
INF3080-1:	Assessment Guide: Information Management Tools Project	G.50
INF3140-1:	Assessment Guide: Artificial Intelligence (AI) Project	G.51
INF3140-2:	Artificial Intelligence (AI) Sample Project	G.52
INF3180-1:	Assessment Checklist: Telecommunication Systems Infrastructure Presentation/Report	G.54
INF3180-2:	Assessment Checklist: Telecommunication Systems Infrastructure Presentation/Report	G.55
INF3180-3:	Assessment Checklist: Telecommunication Design Project	G.56
INF3190-1:	Assessment Guide: Information Highway 3	G.57
INF3200-1:	Assessment Guide: Internet Services	G.58
INF1210-1:	Assessment Checklist: Computer Science 1	G.59
INF1210-2:	Sample Assignment: Computer Science 1	G.61
INF2220-1:	Assessment Checklist: Object-oriented Programming 1	G.63
INF2220-2:	Sample Assignments: Object-oriented Programming 1	G.64
INF2210-1:	Assessment Checklist: Computer Science 2	G.66
INF2210-2:	Sample Assignment: Computer Science 2	G.68
INF3220-1:	Assessment Checklist: Object-oriented Programming 2	G.70
INF3220-2:	Sample Assignment: Object-oriented Programming 2	G.71
INF3230-1:	Assessment Checklist: Dynamic Data Structures 1	G.73
INF3230-2:	Sample Assignment: Dynamic Data Structures 1	G.74
INF3240-1:	Assessment Checklist: Dynamic Data Structures 2	G.75
INF3240-2:	Sample Assignment: Dynamic Data Structures 2	G.76
INF3210-1:	Assessment Checklist: Computer Science 3	G.77
INF3210-2:	Sample Assignment: Computer Science 3	G.79

ASSESSMENT CHECKLIST: COMPUTER SCIENCE 1

INF1210-1

STUDENT: _____

STANDARD:	Students working at standard must demonstrate a general understanding of the nature, approaches, areas of interest and algorithmic basis of the discipline of computer science. As the main focus of computer science is the utilization of algorithmic approaches to problem solving, most of the emphasis of this course is on the ability to understand design, develop, implement and test algorithmic solutions to problems amenable to structured programming approaches. As this course is designed to be taught in conjunction with Programming 1 and Programming 2 this assessment checklist dovetails with the checklist for those courses. The column to the left of each checklist indicates the minimum rating for at standard performance. The rating scale at the bottom defines the different levels of competencies.
------------------	---

At Standard	Computer Science 1	
2	<p>Background Knowledge and Skills: The student:</p> <ul style="list-style-type: none"> <input type="checkbox"/> demonstrates an introductory understanding of the nature, approaches and areas of interest of computer science <input type="checkbox"/> demonstrates an understanding of the nature of the algorithm and the ability to use IPO analysis and design techniques to construct algorithms and programs that reflect the structured programming paradigm 	<ul style="list-style-type: none"> <input type="checkbox"/> demonstrates the ability to illustrate how computer scientists use computer languages to convert an algorithm into a form that can be executed by a computer <input type="checkbox"/> demonstrates through the use of block diagrams, or other appropriate techniques, how a computer uses a program to accept data from the input section, use the CPU and memory to process the data and the output section to display the processed data
2	<p>Utilization of Knowledge and Skills: Design/Development Stage: The student designs an algorithmic solution to a problem that:</p> <ul style="list-style-type: none"> <input type="checkbox"/> demonstrates the ability to analyze a problem and identify the initial state, the final state and the input, output and processing requirements of the problem <input type="checkbox"/> identifies the appropriate literals, constants and variables needed to accommodate the input, output and processing data to be handled in the problem solution <input type="checkbox"/> identifies any idioms required by the program <input type="checkbox"/> identifies the structured programming constructs or control structures needed by the program (sequential, selection and repetitive constructs) 	<ul style="list-style-type: none"> <input type="checkbox"/> demonstrates the ability to design a user interface that prompts for and accepts input, and displays the output in an appropriate format <input type="checkbox"/> creates an algorithm, using an appropriate form (flowchart, pseudo-code, IPO chart) that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program <input type="checkbox"/> passes the "fail on paper" test
2	<p>Utilization of Knowledge and Skills: Implementation Phase: The student converts an algorithm into a program that requires the following:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Initialization <ul style="list-style-type: none"> • introduces program to user giving purpose and process • introduces user to interface • declares and initializes constants and variables identified in the algorithm <input type="checkbox"/> Input <ul style="list-style-type: none"> • prompts user for input • accepts input • uses appropriate decision and iterative constructs to check input (type and range) <input type="checkbox"/> Processing <ul style="list-style-type: none"> • updates variables to reflect input • carries out calculations, comparisons, incrementing required by the updating • uses appropriate sequential, selection and repetitive constructs to process data • updates information to be displayed to user 	<ul style="list-style-type: none"> <input type="checkbox"/> Output <ul style="list-style-type: none"> • displays updated information to user in appropriate format • highlights significant information <input type="checkbox"/> Linking/Termination <ul style="list-style-type: none"> • determines if a final state has been achieved • if a final state has not been achieved, loop back to input to repeat the input processing/output cycle • if a final state has been achieved, terminate the program in an appropriate manner

2	<p>Utilization of Knowledge and Skills: Execution Phase: The student tests and completes the documentation of a program created earlier by the student from an algorithm. These processes should do the following:</p> <div><div><p><input type="checkbox"/> Testing</p><ul style="list-style-type: none">• program is tested with data that produces known results• program is tested with boundary data to test for OB1 (Off By 1) errors• program is tested with aberrant data to test error checking• all necessary modifications are made</div><div><p><input type="checkbox"/> Documentation</p><ul style="list-style-type: none">• does internal and external documentation through all stages employing an appropriate reporting approach• presents statement of problem and algorithm to show problem solution• outlines scope or limits of the program solution• presents documented code listings</div></div>				
	Rating Scale	4 – Demonstrates initiative that exceeds required knowledge/techniques/skills.	3 – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	2 – Demonstrates all designated techniques/skills; occasionally needs prompting.	1 – Demonstrates most designated techniques/skills; frequently needs prompting.

An Ecology Simulation

A biology teacher at your school has approached you to write a simple ecology simulation for an introductory science class. The ecology is to have the following characteristics:

- the setting for the ecology is to be a closed system, such as an island
- the ecology is to have one type of plant life, one type of herbivore and one type of carnivore
- the students (the users) will be allowed to set the size of the island (in hectares), the initial number of plants, the initial number of herbivores and the initial number of carnivores
- the simulation is to run in yearly cycles; during each year, the carnivores are to prey on the herbivores and the herbivores are to feed on the plant life. In addition, each species is to reproduce over the course of the year
- each carnivore requires 50 herbivores a year to survive, each herbivore requires 5000 plants a year to survive and each hectare of land can support up to 100 000 plants
- the carnivores' rate of reproduction is 1 to 2 (i.e., 1 carnivore can produce 2 offspring—assuming that there are a minimum of 2 carnivores in the system), the herbivores' rate of reproduction is 1 to 6 and the plants' rate of reproduction is 1 to 50
- assume that each species only consumes what it needs to survive (i.e., that the carnivores only eat 50 herbivores a year), that plants and animals that cannot get the food they need to survive die, that all calculations are done at the end of each year and that predation occurs prior to reproduction
- run the simulation for 10 years, or until the user wishes to exit or until the ecology “crashes”; (This ecology can be said to have crashed when one or more of the species dies out. Note: Simple ecologies are very unstable and are prone to crashing.)
- display the results of each cycle as a row in a table. As students are not expected to be familiar with arrays and/or vectors, this data will have to be displayed as the simulation is executing with each “year’s” data being displayed immediately after it is calculated.

Design and develop an algorithm that:

- employs problem parsing to analyze the problem
- uses the expanded IPO paradigm to structure the algorithm
- uses the top-down, step-wise refinement approach to add detail to the algorithm
- uses an appropriate nomenclature, such as flowcharting or pseudo-code
- sketches a screen display(s) that illustrates the user interface outlined in the algorithm
- passes a walk-through or failed-on-paper test.

Translate the algorithm into an executable program that:

- maintains the logic and structure of the algorithm
- employs good structured programming practices (structured constructs and blocks)
- incorporates the user interface designed in the analysis and design stages
- is built in executable increments a few statements at a time
- has adequate internal and external documentation.

Test and implement the program by:

- executing the program with data known to produce specific output
- executing the program with aberrant data
- checking for congruency with the original requirements of the problem.

For assessment standards and criteria, see Assessment Checklist: Computer Science 1, INF1210-1.

ASSESSMENT CHECKLIST: OBJECT-ORIENTED PROGRAMMING 1

INF2220-1

STUDENT: _____

STANDARD	Students working at standard must demonstrate use of problem-solving techniques when producing programs, using criteria as noted in the checklists below. The column to the left of each checklist indicates the minimum rating for at standard performance. The rating scale at the bottom defines the different levels of competencies.
----------	---

At Standard	<i>Object-oriented Programming 1</i>	
2	<p>Problem-solving Phase: The student:</p> <ul style="list-style-type: none"> <input type="checkbox"/> defines the nature of the problem and outlines what the program must do <input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program <input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures) and classes required in the program <input type="checkbox"/> codes the algorithm, using a programming language 	<ul style="list-style-type: none"> <input type="checkbox"/> documents comments to programmers <input type="checkbox"/> debugs and tests sample data <input type="checkbox"/> codes and formats the program properly <input type="checkbox"/> evaluates the final product to insure proper implementation (see below)
3	<p>Implementation Phase: The student creates a minimum of three programs containing the following—see sample assignments for Object-oriented Programming 1:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Initialization and Input <ul style="list-style-type: none"> • stringed, integer and real variables • numeric and string constants • data entered through assignment statements and keyboard entry • appropriate local and global variables • data is stored in appropriate derived data types • error trapping occurs, using appropriate derived data types • data components of a class are identified (object oriented only) • modification of existing classes • construction of new classes • creation of class libraries • instantiation of new objects • data stored in appropriate class data members <input type="checkbox"/> Processes <ul style="list-style-type: none"> • addition, subtraction, multiplication, division • predetermine, pre-check and post-check looping constructs • decision-making constructs • appropriate subprogram structures • proper one- and two-way parameter passing • summation of data stored in arrays 	<ul style="list-style-type: none"> <input type="checkbox"/> Processes (continued) <ul style="list-style-type: none"> • predefined string functions and procedures • methods to be used in classes are identified (object oriented only) • objects are constructed, employing user-defined classes (object oriented only) • data is transferred to objects (object oriented only) • classes are accessed from class libraries • data is updated and transferred to and from objects • class members functions are used for processing <input type="checkbox"/> Output and Termination <ul style="list-style-type: none"> • rounds to a prescribed number of decimal places • lines up decimal points and inserts dollar signs where appropriate • column formatting occurs • data is output from class data members <input type="checkbox"/> Documentation and Presentation <ul style="list-style-type: none"> • presents statement of problem and algorithm to show how program was created • presents user's guide with clear and concise instructions • describes problems encountered during production and testing • aesthetic presentation: uses acceptable design principles • class relationship diagrammed

Rating Scale	4 – Demonstrates initiative that exceeds required techniques/skills.	3 – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	2 – Demonstrates all designated techniques/skills; occasionally needs prompting.	1 – Demonstrates most designated techniques/skills; frequently needs prompting.	0 – Does not demonstrate designated techniques/skills.
--------------	--	---	--	---	--

Your school employs a number of students to run the copy machines. The employees are paid \$5.00 per hour. The tax rate varies according to the amount earned—more than \$200.00 per week is calculated at 42%, more than \$100.00 and less than or equal to \$200.00 is calculated at 30%, and less than or equal to \$100.00 pays no tax. Overtime is paid to employees—time and a half to those working over 40 hours per week.

ASSIGNMENT A

Design and code a program that uses an instructor-supplied class (`EmployeeClass`) to store the data and calculate the wages for three employees for a week. The member functions or methods of the class should:

- gather demographic data on each employee—surname, first name and employee number
- store the number of hours worked per week
- calculate the gross pay and deductions, and return both to the main program.

These values should be passed to a subprogram(s) in the client program that prints the hours worked, gross pay, deductions and net pay. The subprogram(s) should include appropriate derived data types for error trapping on data entry. The subprogram(s) should produce the following output:

Individual Employee Reports for (your school)

Employee #1

Name: Harry Smith
Hours Worked: 40
Gross Pay: 200.00
Deductions: 66.67
Net Pay: 133.33

Employee #2

Name: Gordon Elliot
Hours Worked: 50
Gross Pay: 275.00
Deductions: 115.50
Net Pay: 160.50

Employee #3

Name: Ken East
Hours Worked: 10
Gross Pay: 50.00
Deductions: 0.00
Net Pay: 50.00

(continued)

Summaries for the data for each employee:

Name	Hours Worked	Total Gross	Total Deductions	Net Pay
Harry Smith	40	200.00	66.67	133.33
Gordon Elliot	50	275.00	115.50	160.50
Ken East	10	50.00	0.00	50.00

Surname first for the data for each employee:

Name	Hours Worked	Total Gross	Total Deductions	Net Pay
Smith, Harry	40	200.00	66.67	133.33
Elliot, Gordon	50	275.00	115.50	160.50
East, Ken	10	50.00	0.00	50.00

School Summary:

Total Gross	Total Deductions	Total Net
525.00	182.17	343.83

For assessment standards and criteria, see Assessment Checklist: Object-oriented Programming 1, INF2220-1.

ASSIGNMENT B

Modify the above design and program so that the subprogram(s) used to output the Individual Employee Reports is (are) added to the instructor-supplied class as (a) member function(s).

Further modify the program by creating a class that uses data from the EmployeeClass to generate the summaries.

For assessment standards and criteria, see Assessment Checklist: Object-oriented Programming 1, INF2220-1.

STUDENT: _____

STANDARD:	<p>Students working at standard must demonstrate an adequate intermediate understanding of the nature, approaches, areas of interest and algorithmic basis of the discipline of computer science. As the main focus of computer science is the utilization of algorithmic approaches to problem solving, most of the emphasis of this course is on the ability to understand design, develop, implement and test algorithmic solutions to problems amenable to introductory modular programming approaches. As this course is designed to be taught in conjunction with Programming 3 and Programming 4, this assessment checklist dovetails with the checklists for those courses. The column to the left of each checklist indicates the minimum rating for at standard performance. The rating scale at the bottom defines the different levels of competencies.</p>
At Standard	<p style="text-align: center;">Computer Science 2</p> <p>Background Knowledge and Skills: The student:</p> <ul style="list-style-type: none"> <input type="checkbox"/> demonstrates an understanding of the qualitative and quantitative trends in the development of computer technology <input type="checkbox"/> demonstrates a growing understanding of algorithmic problem solving by adding approaches, such as problem decomposition—into subprograms—to the simpler top-down and step-wise refinement problem parsing techniques developed at earlier levels. The student now constructs algorithms and programs that reflect both the structured and the modular programming paradigms <input type="checkbox"/> demonstrates an understanding of how machines employing the von Neumann computer architecture are programmed to process stored data at the machine language level. This is likely best done through the manipulation of a virtual computer with its own “toy” machine language or Assembly language. This virtual computer could either be a computer simulation of a simple computer, a non-computer simulation of a simple computer or a paper-and-pencil depiction of a simple computer <p>Utilization of Knowledge and Skills: Design/Development Stage: The student designs an algorithmic solution to a problem that employs the tactics developed in the Computer Science 1 course and in addition:</p> <ul style="list-style-type: none"> <input type="checkbox"/> uses appropriate subprograms—procedures and functions—with a high level of cohesion and relatively low level of coupling <input type="checkbox"/> uses an appropriate construction technique, such as structure diagrams, HIPO charting, structured charting or Warnier–Orr diagrams to identify levels of abstraction, scope and coupling considerations
2	<p>Utilization of Knowledge and Skills: Implementation Phase: The student converts an algorithm into a program that employs all of the tactics developed in the Computer Science 1 course and in addition should:</p> <ul style="list-style-type: none"> <input type="checkbox"/> utilize subprograms, such as functions and procedures: <ul style="list-style-type: none"> • to create a modular program • to create functionally abstract blocks of code <input type="checkbox"/> utilize appropriate code building approaches, such as: <ul style="list-style-type: none"> • stub programming techniques or its equivalent • iterative prototyping • creation of specialized libraries to promote information hiding <input type="checkbox"/> utilize derived or structured data types, such as arrays, vectors, structs and records to: <ul style="list-style-type: none"> • input, process and output related, structured or matrix data

2	<p>Utilization of Knowledge and Skills: Execution Phase: The student tests and completes the documentation of a program created earlier by the student from an algorithm. These processes should employ all of the tactics developed in the Computer Science 1 course and in addition:</p> <p><input type="checkbox"/> Testing</p> <ul style="list-style-type: none"> • beta tests the program with knowledgeable users (another class member) <p><input type="checkbox"/> Implementation</p> <ul style="list-style-type: none"> • program is presented to end user with appropriate instructions <p><input type="checkbox"/> Maintenance</p> <ul style="list-style-type: none"> • user feedback is solicited to guide future revisions <p><input type="checkbox"/> Documentation</p> <ul style="list-style-type: none"> • describes problems encountered during design, development, production and testing • describes possible updates based on user feedback
Rating Scale	<p>4 – Demonstrates initiative that exceeds required knowledge/techniques/skills.</p> <p>3 – Consistently demonstrates all designated techniques/skills; rarely needs prompting.</p> <p>2 – Demonstrates all designated techniques/skills; occasionally needs prompting.</p> <p>1 – Demonstrates most designated techniques/skills; frequently needs prompting.</p> <p>0 – Does not demonstrate designated techniques/skills.</p>

An Ecology Simulation (An enhanced version of the sample assignment outlined in INF_CSSAMP1)

A biology teacher at your school has approached you to write a simple ecology simulation for an introductory science class. The ecology is to have the following characteristics:

- the setting for the ecology is to be a closed system, such as an island
- the ecology is to have one type of plant life, one type of herbivore and one type of carnivore
- the students (the users) will be allowed to set the size of the island (in hectares), the initial number of plants, the initial number of herbivores and the initial number of carnivores
- the simulation is to run in yearly cycles; during each year, the carnivores are to prey on the herbivores and the herbivores are to feed on the plant life. In addition, each species is to reproduce over the course of the year. The processes of predation and reproduction are ideally suited to be handled as functional abstractions through the use of subproblems and subprograms
- each carnivore requires 50 herbivores a year to survive, each herbivore requires 5000 plants a year to survive and each hectare of land can support up to 100 000 plants
- the carnivores' rate of reproduction is 1 to 2 (i.e., 1 carnivore can produce 2 offspring—assuming that there are a minimum of 2 carnivores in the system), the herbivores' rate of reproduction is 1 to 6 and the plants' rate of reproduction is 1 to 50
- assume that each species only consumes what it needs to survive (i.e., that the carnivores only eat 50 herbivores a year), that plants and animals that can not get the food they need to survive die, that all calculations are done at the end of each year and that predation occurs prior to reproduction
- run the simulation for 10 years, or until the user wishes to exit or until the ecology “crashes”: (This ecology can be said to have crashed when one or more of the species dies out. Note: Simple ecologies are very unstable and are prone to crashing.)
- display the results of each cycle as a row in a table. As students are expected to be familiar with arrays and/or vectors, this data should be stored in an appropriate derived data type, such as an array, vector, record or struct and recalled when needed
- display the results as either a line graph or bar chart. The data for this display would have to be extracted from a derived data type, such as an array.

Design and develop an algorithm that:

- employs problem parsing to analyze the problem and decompose it into a hierarchy of subproblems
- identifies similar subproblems that might be amenable to reusable code
- uses the expanded IPO paradigm to structure the first level of the algorithm
- uses a top-down, step-wise refinement approach to add detail to each level of subproblem in the algorithm
- uses an appropriate nomenclature, such as HIPO or structure charts, Warnier–Orr diagrams, or pseudo-code to outline the algorithm
- incorporates an appropriate data dictionary and outlines the required data structures
- sketches a screen display(s) that illustrates the user interface and the format of the graphical output
- passes a walk-through or failed-on-paper test.

Translate the algorithm into an executable program that:

- maintains the logic and structure of the algorithm
- employs good modular programming practices (loosely coupled, highly cohesive subprograms with extensive use of local data)
- incorporates the user interface and displays designed in the analysis and design stages
- uses a technique, such as stub programming to translate the various levels of the algorithm into executable code:
 - uses top-down coding to build a program in executable increments a few statements at a time
 - uses bottom-up coding to build subprograms for testing
 - makes appropriate use of derived data types to input, process and output information
- has adequate internal and external documentation.

Test and implement the program by:

- executing the program with data known to produce specific output
- executing the program with aberrant data
- checking for congruency with the original requirements of the problem
- beta testing the program with knowledgeable users, such as another class member
- providing adequate end user instructions
- soliciting user feedback to guide future revisions.

For assessment standards and criteria, see Assessment Checklist: Computer Science 2, INF2210-1.

STUDENT: _____

STANDARD	Students working at standard must demonstrate use of problem-solving techniques when producing programs, using criteria as noted in the checklists below. The column to the left of each checklist indicates the minimum rating for at standard performance. The rating scale at the bottom defines the different levels of competencies.	
At Standard	Object-oriented Programming 2	
2	<p>Problem-solving Phase: The student:</p> <ul style="list-style-type: none"> <input type="checkbox"/> defines the nature of the problem and outlines what the program must do <input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program <input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program 	<ul style="list-style-type: none"> <input type="checkbox"/> codes the algorithm, using a programming language <input type="checkbox"/> documents comments to programmers <input type="checkbox"/> debugs and tests sample data <input type="checkbox"/> codes and formats the program properly <input type="checkbox"/> evaluates the final product to insure proper implementation (see below)
3	<p>Implementation Phase: The student creates a minimum of three programs containing the following—see sample assignment for Object-oriented Programming 2:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Initialization and Input <ul style="list-style-type: none"> • character, string, integer, real, Boolean and user-defined variables • numeric, character, string and parameter constants • data entered through keyboard entry and assignment statements • appropriate use of scope—local and global variables • data is stored in appropriate derived data types and class data members • error trapping occurs, using appropriate approaches, including derived data types and class member functions • modification of existing classes • construction of new classes • creation of class libraries • instantiation of new objects • data stored in appropriate class data members • construction of templated classes • appropriate use of composition or containment • construction of base and derived classes • use of constructor and operator overloading <input type="checkbox"/> Processes <ul style="list-style-type: none"> • addition, subtraction, multiplication, division, absolute numbers, exponentiation • truncates or rounds to a prescribed number of decimal places • decision-making constructs • iterative and recursive constructs • predetermine, pre-check and post-check looping constructs <input type="checkbox"/> Documentation and Presentation <ul style="list-style-type: none"> • presents statement of problem and algorithm to show how program was created • presents user's guide with clear and concise instructions • describes problems encountered during production and testing • aesthetic presentation: uses acceptable design principles 	<p>Processes (continued)</p> <ul style="list-style-type: none"> • use of accessors and modifiers to control class access • appropriate use of predefined and user-defined functions and/or procedures • proper return of data, and one- and two-way parameter passing in subprograms • data processing, such as summation in derived data structures; e.g., arrays • classes are accessed from class libraries • data is updated and transferred to and from objects • class members functions are used for processing • codes are profiled and optimized for delivery <p>Output and Termination</p> <ul style="list-style-type: none"> • rounds to a prescribed number of decimal places • data is output from class data members • formats output in an appropriate manner; i.e., <ul style="list-style-type: none"> – places text at specified locations on the screen – lines up decimal points and inserts dollar signs – column formatting occurs • uses destructors to terminate objects
Rating Scale	<p>4 – Demonstrates initiative that exceeds required techniques/skills.</p> <p>3 – Consistently demonstrates all designated techniques/skills; rarely needs prompting.</p> <p>2 – Demonstrates all designated techniques/skills; occasionally needs prompting.</p> <p>1 – Demonstrates most designated techniques/skills; frequently needs prompting.</p> <p>0 – Does not demonstrate designated techniques/skills.</p>	

Your school employs a number of students to run the copy machines. The employees are paid a base rate of \$5.00 per hour. The tax rate varies according to the amount earned—more than \$200.00 per week is calculated at 42%, more than \$100.00 and less than or equal to \$200.00 is calculated at 30%, and less than or equal to \$100.00 pays no tax. Overtime is paid to employees—time and a half to those working over 40 hours per week.

Your school's duplicating department has grown to the point where it is taking in work from other schools. To accommodate this extra work, two students have been promoted to employee/manager. Their base rate of pay is \$8.00 per hour. In addition, they get a 10% increase on their base rate of pay for every hour (or part hour) that they supervise three or more students.

Design and code a program that uses a class (EmployeeClass) to store the data and to calculate wages for three regular employees for a week. The member functions or methods of the class should:

- gather demographic data on each employee—surname, first name and employee number
- store the number of hours worked per week over the course of the month—four weeks equals one month
- calculate the gross pay and deductions, and return both to the main program.

These values should be passed to a subprogram(s) in the client program that prints the hours worked, gross pay, deductions and net pay. The subprogram(s) should include appropriate derived data types for error trapping on data entry. The subprogram(s) should produce the following output:

Individual Employee Reports for (your school)

Employee #1	Employee #2	Employee #3
Name: Harry Smith Hours Worked: 40 Gross Pay: 200.00 Deductions: 66.67 Net Pay: 133.33	Name: Gordon Elliot Hours Worked: 50 Gross Pay: 275.00 Deductions: 115.50 Net Pay: 160.50	Name: Ken East Hours Worked: 10 Gross Pay: 50.00 Deductions: 0.00 Net Pay: 50.00

(continued)

Construct a derived class based on `EmployeeClass` to store the data and to calculate wages for the two employee/managers for a month. The member functions or methods of this derived class should:

- gather demographic data on each employee/manager—surname, first name and employee/manager number
- store the number of hours worked per week
- store the number of hours the employee/manager supervised three or more employees
- calculate the regular pay, supervisor pay and deductions; and return these values to the client program.

These values should be passed to a subprogram(s) in the client program that prints the hours worked, the regular and supervisory pay, the deductions and the net pay. The subprogram(s) should include appropriate derived data types for error trapping on data entry. The subprogram(s) should produce the following output:

Individual Employee/Manager Reports for (your school)

Employee/Manager #1

Name: Sam Handwich
Regular Hours: 30
Supervisory Hours: 5
Regular Pay: 240.00
Supervisory Pay: 44.00
Deductions: 119.28
Net Pay: 164.72

Employee/Manager #2

Name: Tam Jart
Regular Hours: 25
Supervisory Hours: 10
Regular Pay: 200.00
Supervisory Pay: 88.00
Deductions: 120.96
Net Pay: 167.04

For assessment standards and criteria, see Assessment Checklist: Object-oriented Programming 2, INF3220-1.

ASSESSMENT CHECKLIST: DYNAMIC DATA STRUCTURES 1

INF3230-1

STUDENT: _____

STANDARD	Students working at standard must demonstrate use of problem-solving techniques when producing programs, using criteria as noted in the checklists below. The column to the left of each checklist indicates the minimum rating for at standard performance. The rating scale at the bottom defines the different levels of competencies.
----------	---

At Standard	<i>Dynamic Data Structures 1</i>	
2	<p>Problem-solving Phase: The student:</p> <ul style="list-style-type: none"> <input type="checkbox"/> defines the nature of the problem and outlines what the program must do <input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program <input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program <input type="checkbox"/> identifies the appropriate use of linked lists required to solve the problem 	<ul style="list-style-type: none"> <input type="checkbox"/> codes the algorithm, using a programming language <input type="checkbox"/> documents comments to programmers <input type="checkbox"/> debugs and tests sample data <input type="checkbox"/> codes and formats the program properly <input type="checkbox"/> evaluates the final product to insure proper implementation (see below)
3	<p>Implementation Phase: The student creates a minimum of three programs containing the following—see sample assignment for Dynamic Data Structures 1:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Initialization and Input <ul style="list-style-type: none"> • stringed, integer and real variables • numeric and string constants • data entered through assignment statements and keyboard entry • appropriate local and global variables • data is stored in appropriate derived data types • error trapping occurs, using appropriate derived data types • data components of a class are identified (object oriented only) • modification to existing classes are identified (object oriented only) • characteristics to be contained or inherited by new classes are identified (object oriented only) • pointers and linked lists are created as required • data is loaded into linked lists as required <input type="checkbox"/> Processes <ul style="list-style-type: none"> • addition, subtraction, multiplication, division • predetermine, pre-check and post-check looping constructs • decision-making constructs • appropriate subprogram structures • proper one- and two-way parameter passing • summation of stored data in arrays • predefined string functions and procedures • methods to be used in classes are identified (object oriented only) • objects are constructed, employing user-defined classes (object oriented only) 	<ul style="list-style-type: none"> <input type="checkbox"/> Processes (continued) <ul style="list-style-type: none"> • data is transferred to objects (object oriented only) • sorting based on differing criteria (object oriented only) • search and merge routines (object oriented only) • dynamic memory is allocated and de-allocated as required • pointers are used to process data and direct program flow • data is processed in linked lists as required • nodes are added and/or deleted in linked lists as required • linked lists are traversed/searched/sorted <input type="checkbox"/> Output and Termination <ul style="list-style-type: none"> • rounds to a prescribed number of decimal places • lines up decimal points and inserts dollar signs where appropriate • column formatting occurs • de-allocates dynamic memory • disposes of linked lists <input type="checkbox"/> Documentation and Presentation <ul style="list-style-type: none"> • presents statement of problem and algorithm to show how program was created • presents user's guide with clear and concise instructions • describes problems encountered during production and testing • aesthetic presentation: uses acceptable design principles

Rating Scale	4 – Demonstrates initiative that exceeds required techniques/skills.	3 – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	2 – Demonstrates all designated techniques/skills; occasionally needs prompting.	1 – Demonstrates most designated techniques/skills; frequently needs prompting.	0 – Does not demonstrate designated techniques/skills.
--------------	--	---	--	---	--

Your school has several copy machines. The staff in the main office use one of the large ones. Normally, jobs on this machine are done in the order they are received. Usually the first job in is the first job done; however, rush jobs can be inserted into the list.

Design and code a program that uses a linked list to keep track of the order in which jobs must be done. Have the program randomly generate between five and ten duplicating jobs. Give each job a submit time and a completion time. Assume that these jobs are all submitted between 9:00 and 10:00 in the morning. Assume that each job takes 30 minutes. As each job is generated, it is added to a linked list that expands to accommodate the number of jobs. Once all jobs are added to the list, ask the user if he/she has any rush jobs that must be added to the list. If the answer is yes, ask the user for a “required by time.” Have the program insert this new job into the list so that it is completed prior to its “required by time” with the minimum amount of “bumping” of jobs already listed. Use “After 10:00” as a submitted time. Use a 24-hour clock.

Your program should generate the following output:

The following jobs were submitted for duplication:

Job 1: Submit Time 9:00	Projected Completion Time 9:30
Job 2: Submit Time 9:05	Projected Completion Time 10:00
Job 3: Submit Time 9:10	Projected Completion Time 10:30
Job 4: Submit Time 9:15	Projected Completion Time 11:00
Job 5: Submit Time 9:20	Projected Completion Time 11:30
Job 6: Submit Time 9:30	Projected Completion Time 12:00

Do you have any rush jobs (Y or N)? Y

What is the required by time for the job (use 24-hour notation)? 10:30

Your job has been assigned number 7.

The jobs were done in the following order:

Job 1: Submit Time 9:00	Projected Completion Time 9:30
Job 2: Submit Time 9:05	Projected Completion Time 10:00
Job 7: Submit Time After 10:00	Projected Completion Time 10:30
Job 3: Submit Time 9:10	Projected Completion Time 11:00
Job 4: Submit Time 9:15	Projected Completion Time 11:30
Job 5: Submit Time 9:20	Projected Completion Time 12:00
Job 6: Submit Time 9:30	Projected Completion Time 12:30

For assessment standards and criteria, see Assessment Checklist: Dynamic Data Structures 1, INF3230-1.

ASSESSMENT CHECKLIST: DYNAMIC DATA STRUCTURES 2

INF3240-1

STUDENT: _____

STANDARD	Students working at standard must demonstrate use of problem-solving techniques when producing programs, using criteria as noted in the checklists below. The column to the left of each checklist indicates the minimum rating for at standard performance. The rating scale at the bottom defines the different levels of competencies.
----------	---

At Standard	<i>Dynamic Data Structures 2</i>
2	<p>Problem-solving Phase: The student:</p> <ul style="list-style-type: none"> <input type="checkbox"/> defines the nature of the problem and outlines what the program must do <input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program <input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program <input type="checkbox"/> identifies the appropriate dynamic data structure required to solve the problem <ul style="list-style-type: none"> <input type="checkbox"/> codes the algorithm, using a programming language <input type="checkbox"/> documents comments to programmers <input type="checkbox"/> debugs and tests sample data <input type="checkbox"/> codes and formats the program properly <input type="checkbox"/> evaluates the final product to insure proper implementation (see below)
3	<p>Implementation Phase: The student creates a minimum of three programs containing the following—see sample assignment for Dynamic Data Structures 2:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Initialization and Input <ul style="list-style-type: none"> • stringed, integer and real variables • numeric and string constants • data entered through assignment statements and keyboard entry • appropriate local and global variables • data is stored in appropriate derived data types • error trapping occurs, using appropriate derived data types • data components of a class are identified (object oriented only) • modification to existing classes are identified (object oriented only) • characters to be inherited by new classes are identified (object oriented only) • pointers, linked lists, stacks, queues and trees are created as required • data is loaded into linked lists, stacks, queues and trees as required <input type="checkbox"/> Processes <ul style="list-style-type: none"> • addition, subtraction, multiplication, division • predetermine, pre-check and post-check looping constructs • decision-making constructs • appropriate subprogram structures • proper one- and two-way parameter passing • summation of stored data in arrays • predefined string functions and procedures • methods to be used in classes are identified (object oriented only) • objects are constructed, employing user-defined classes (object oriented only) <ul style="list-style-type: none"> <input type="checkbox"/> Processes (continued) <ul style="list-style-type: none"> • data is transferred to objects (object oriented only) • sorting based on differing criteria (object oriented only) • search and merge routines (object oriented only) • dynamic memory is allocated and de-allocated as required • pointers are used to process data and direct program flow • push and pop data • enqueue and dequeue data • data is processed in linked lists, stacks, queues and trees as required • nodes are added and/or deleted in linked lists and trees as required • linked lists and trees are traversed/searched/sorted <input type="checkbox"/> Output and Termination <ul style="list-style-type: none"> • rounds to a prescribed number of decimal places • lines up decimal points and inserts dollar signs where appropriate • column formatting occurs • de-allocates dynamic memory • disposes of linked lists <input type="checkbox"/> Documentation and Presentation <ul style="list-style-type: none"> • presents statement of problem and algorithm to show how program was created • presents user's guide with clear and concise instructions • describes problems encountered during production and testing • aesthetic presentation: uses acceptable design principles

Rating Scale	4 – Demonstrates initiative that exceeds required techniques/skills.	3 – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	2 – Demonstrates all designated techniques/skills; occasionally needs prompting.	1 – Demonstrates most designated techniques/skills; frequently needs prompting.	0 – Does not demonstrate designated techniques/skills.
--------------	--	---	--	---	--

Your school has several copy machines. The staff in the main office use one of the large ones. Normally, jobs on this machine are queued. Usually the first job in is the first job done; however, jobs can be prioritized. Rush jobs are done before ordinary jobs.

Design and code a program that uses a linked list to be used as a queue. Have the program randomly generate between five and ten duplicating jobs; give each job a submit time and a priority. As each job is generated, it is added to a linked list that expands to accommodate the number of jobs. Once all jobs are added to the queue, the program reorders the list so that rush jobs are done before ordinary jobs. Rush jobs are done in the order of submission. Use a 24-hour clock.

Your program should generate the following output:

The following jobs were submitted for duplication:

Job 1:	Submit Time	9:00	Priority Normal
Job 2:	Submit Time	9:10	Priority Rush
Job 3:	Submit Time	10:20	Priority Normal
Job 4:	Submit Time	11:10	Priority Normal
Job 5:	Submit Time	12:00	Priority Rush
Job 6:	Submit Time	13:20	Priority Normal

The jobs were done in the following order:

Job 2:	Submit Time	9:10	Priority Rush
Job 5:	Submit Time	12:00	Priority Rush
Job 1:	Submit Time	9:00	Priority Normal
Job 3:	Submit Time	10:20	Priority Normal
Job 4:	Submit Time	11:10	Priority Normal
Job 6:	Submit Time	13:20	Priority Normal

For assessment standards and criteria, see Assessment Checklist: Dynamic Data Structures 2, INF3240-1.

STUDENT: _____

STANDARD	Students working at standard must demonstrate use of problem-solving techniques when producing programs, using criteria as noted in the checklists below. The column to the left of each checklist indicates the minimum rating for at standard performance. The rating scale at the bottom defines the different levels of competencies.	
At Standard	Dynamic Data Structures 2	
2	<p>Problem-solving Phase:</p> <p>The student:</p> <ul style="list-style-type: none"> <input type="checkbox"/> defines the nature of the problem and outlines what the program must do <input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program <input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program <input type="checkbox"/> identifies the appropriate dynamic data structure required to solve the problem 	<ul style="list-style-type: none"> <input type="checkbox"/> codes the algorithm, using a programming language <input type="checkbox"/> documents comments to programmers <input type="checkbox"/> debugs and tests sample data <input type="checkbox"/> codes and formats the program properly <input type="checkbox"/> evaluates the final product to insure proper implementation (see below)
3	<p>Implementation Phase: The student creates a minimum of three programs containing the following—see sample assignment for Dynamic Data Structures 2:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Initialization and Input <ul style="list-style-type: none"> • stringed, integer and real variables • numeric and string constants • data entered through assignment statements and keyboard entry • appropriate local and global variables • data is stored in appropriate derived data types • error trapping occurs, using appropriate derived data types • data components of a class are identified (object oriented only) • modification to existing classes are identified (object oriented only) • characters to be inherited by new classes are identified (object oriented only) • pointers, linked lists, stacks, queues and trees are created as required • data is loaded into linked lists, stacks, queues and trees as required <input type="checkbox"/> Processes <ul style="list-style-type: none"> • addition, subtraction, multiplication, division • predetermine, pre-check and post-check looping constructs • decision-making constructs • appropriate subprogram structures • proper one- and two-way parameter passing • summation of stored data in arrays • predefined string functions and procedures • methods to be used in classes are identified (object oriented only) • objects are constructed, employing user-defined classes (object oriented only) 	<ul style="list-style-type: none"> <input type="checkbox"/> Processes (continued) <ul style="list-style-type: none"> • data is transferred to objects (object oriented only) • sorting based on differing criteria (object oriented only) • search and merge routines (object oriented only) • dynamic memory is allocated and de-allocated as required • pointers are used to process data and direct program flow • push and pop data • enqueue and dequeue data • data is processed in linked lists, stacks, queues and trees as required • nodes are added and/or deleted in linked lists and trees as required • linked lists and trees are traversed/searched/sorted <input type="checkbox"/> Output and Termination <ul style="list-style-type: none"> • rounds to a prescribed number of decimal places • lines up decimal points and inserts dollar signs where appropriate • column formatting occurs • de-allocates dynamic memory • disposes of linked lists <input type="checkbox"/> Documentation and Presentation <ul style="list-style-type: none"> • presents statement of problem and algorithm to show how program was created • presents user's guide with clear and concise instructions • describes problems encountered during production and testing • aesthetic presentation: uses acceptable design principles

Rating Scale	4 – Demonstrates initiative that exceeds required techniques/skills.	3 – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	2 – Demonstrates all designated techniques/skills; occasionally needs prompting.	1 – Demonstrates most designated techniques/skills; frequently needs prompting.	0 – Does not demonstrate designated techniques/skills.
--------------	--	---	--	---	--

Your school has several copy machines. The staff in the main office use one of the large ones. Normally, jobs on this machine are queued. Usually the first job in is the first job done; however, jobs can be prioritized. Rush jobs are done before ordinary jobs.

Design and code a program that uses a linked list to be used as a queue. Have the program randomly generate between five and ten duplicating jobs; give each job a submit time and a priority. As each job is generated, it is added to a linked list that expands to accommodate the number of jobs. Once all jobs are added to the queue, the program reorders the list so that rush jobs are done before ordinary jobs. Rush jobs are done in the order of submission. Use a 24-hour clock.

Your program should generate the following output:

The following jobs were submitted for duplication:

Job 1:	Submit Time	9:00	Priority	Normal
Job 2:	Submit Time	9:10	Priority	Rush
Job 3:	Submit Time	10:20	Priority	Normal
Job 4:	Submit Time	11:10	Priority	Normal
Job 5:	Submit Time	12:00	Priority	Rush
Job 6:	Submit Time	13:20	Priority	Normal

The jobs were done in the following order:

Job 2:	Submit Time	9:10	Priority	Rush
Job 5:	Submit Time	12:00	Priority	Rush
Job 1:	Submit Time	9:00	Priority	Normal
Job 3:	Submit Time	10:20	Priority	Normal
Job 4:	Submit Time	11:10	Priority	Normal
Job 6:	Submit Time	13:20	Priority	Normal

For assessment standards and criteria, see Assessment Checklist: Dynamic Data Structures 2, INF3240-1.

STUDENT: _____

STANDARD:

Students working at standard must demonstrate a post-secondary entry level understanding of the nature, approaches, areas of interest and algorithmic basis of the discipline of computer science. As the main focus of computer science is the utilization of algorithmic approaches to problem solving, most of the emphasis of this course is on the ability to understand design, develop, implement and test algorithmic solutions to problems amenable to intermediate modular and introductory object-oriented programming approaches. As this course is designed to be taught in conjunction with Programming 5 and Object-oriented Programming 1, this assessment checklist dovetails with the checklists for those courses. The column to the left of each checklist indicates the minimum rating for at standard performance. The rating scale at the bottom defines the different levels of competencies.

At Standard	
2	<p>Computer Science 3</p> <p>Background Knowledge and Skills: The student:</p> <ul style="list-style-type: none"> <input type="checkbox"/> demonstrates an understanding of the historical roots, processes, trends and general nature of the information revolution and the emerging information society <input type="checkbox"/> demonstrates a growing understanding of algorithmic problem solving by adding object-oriented design and programming approaches, such as object identification, analysis, creation and manipulation to the structured and modular approaches developed at earlier levels. The student now constructs algorithms and programs that reflect the OOD/OOP paradigm, which in turn employs structured and modular approaches <input type="checkbox"/> demonstrates a general knowledge of core OOD/OOP concepts, such as encapsulation, inheritance and polymorphism <input type="checkbox"/> demonstrates a general understanding of the basic nature, advantages and disadvantages of recursion and recursive processes <ul style="list-style-type: none"> <input type="checkbox"/> demonstrates a general understanding of the basic nature, creation and utility of different types of files <input type="checkbox"/> demonstrates a general understanding of the basic nature, creation and utility of Abstract Data Types and an ability to use both simpler data structures and classes to create Abstract Data Structures <input type="checkbox"/> demonstrates an understanding of how a Turing machine can be used as a model of a general computing agent and used to explore the nature of problem analysis. As part of this process, students would construct and execute a number of standard algorithms such as a bit inverter or a parity checker on a Turing machine. This would likely best be done through the manipulation of an actual Turing machine. This machine could either be a computer simulation, a non-computer simulation or a paper-and-pencil depiction of a Turing machine
2	<p>Utilization of Knowledge and Skills: Design/Development Stage: The student designs an algorithmic solution to a problem that employs the tactics developed in the Computer Science 1 and Computer Science 2 courses and in addition:</p> <ul style="list-style-type: none"> <input type="checkbox"/> demonstrates the ability to use a simplified requirement analysis to cast a problem into a system of interacting objects <input type="checkbox"/> demonstrates the ability to use appropriate class design approaches, such as iterative approaches to conceptualize these objects as composed of member functions or methods and data members or properties <input type="checkbox"/> uses modular and structured approaches to outline the logic and structure of these object members employing program decomposition to the point where known idioms can be employed <ul style="list-style-type: none"> <input type="checkbox"/> addresses issues of abstraction, encapsulation and data hiding <input type="checkbox"/> incorporates simple recursive approaches where appropriate <input type="checkbox"/> incorporates data warehousing techniques through the use of files <input type="checkbox"/> uses an appropriate construction technique, such as a simplified version of UML to create the class, object and activity diagrams needed to state the behaviour and properties of each object and the interaction among the objects
2	<p>Utilization of Knowledge and Skills: Implementation Phase: The student converts an algorithm into a program that employs all of the tactics developed in the Computer Science 1 and Computer Science 2 courses and in addition should:</p> <ul style="list-style-type: none"> <input type="checkbox"/> use a coding approach, such as iterative prototyping or the recursive/parallel approach that creates an initial core of functionality that is tested before being expanded to the next level. Ultimately this code/test cycle implements the entire algorithm <input type="checkbox"/> encapsulate the properties and behaviours of the abstractions outlined in the algorithm into well structured classes and objects (ADTs) <input type="checkbox"/> create and elaborates class hierarchies and libraries <ul style="list-style-type: none"> <input type="checkbox"/> create appropriate messaging mechanisms between client and server objects <input type="checkbox"/> implement user interface based on objects <input type="checkbox"/> use simple recursive approaches where appropriate <input type="checkbox"/> incorporate appropriate file creation, reading, manipulation and writing techniques where required

2	<p>Utilization of Knowledge and Skills: Execution Phase: The student tests and completes the documentation of a program created earlier by the student from an algorithm. These processes should employ all of the tactics developed in the Computer Science 1 and Computer Science 2 courses and in addition:</p> <div><div><input type="checkbox"/> Testing<ul style="list-style-type: none">• program is installed, configured and executed on a workstation with a different directory structure than the workstation used to code the program</div><div><input type="checkbox"/> Implementation<ul style="list-style-type: none">• end user is given appropriate installation and configuration as well as execution instructions</div><div><input type="checkbox"/> Maintenance<ul style="list-style-type: none">• user feedback is used to identify, design and code a revision to the program</div><div><input type="checkbox"/> Documentation<ul style="list-style-type: none">• documents the contents of any class libraries to the point where a different programmer could use them in his or her program</div></div>				
	Rating Scale	4 – Demonstrates initiative that exceeds required knowledge/techniques/skills.	3 – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	2 – Demonstrates all designated techniques/skills; occasionally needs prompting.	1 – Demonstrates most designated techniques/skills; frequently needs prompting.

An Ecology Simulation (An enhanced version of the sample assignment outlined in INF_CSSAMP1 and INF_CSSAMP2)

A biology teacher at your school has approached you to write a simple ecology simulation for an introductory science class. The ecology is to have the following characteristics:

- the setting for the ecology is to be a closed system, such as an island
- the ecology is to have one type of plant life, one type of herbivore and one type of carnivore
- the students (the users) will be allowed to set the size of the island (in hectares), the initial number of plants, the initial number of herbivores and the initial number of carnivores
- the simulation is to run in yearly cycles; during each year, the carnivores are to prey on the herbivores and the herbivores are to feed on the plant life. In addition, each species is to reproduce over the course of the year
- each carnivore requires 50 herbivores a year to survive, each herbivore requires 5000 plants a year to survive and each hectare of land can support up to 100 000 plants
- the carnivores' rate of reproduction is 1 to 2 (i.e., 1 carnivore can produce 2 offspring—assuming that there are a minimum of 2 carnivores in the system), the herbivores' rate of reproduction is 1 to 6 and the plants' rate of reproduction is 1 to 50
- assume that each species only consumes what it needs to survive (i.e., that the carnivores only eat 50 herbivores a year), that plants and animals that cannot get the food they need to survive die, that all calculations are done at the end of each year and that predation occurs prior to reproduction
- run the simulation for 10 years, or until the user wishes to exit or until the ecology “crashes”. (This ecology can be said to have crashed when one or more of the species dies out. Note: Simple ecologies are very unstable and are prone to crashing.)
- display the results of each cycle as a row in a table. As students are expected to be familiar with arrays and/or vectors, this data should be stored in an appropriate derived data type, such as an array, vector, record or struct and recalled when needed
- store the data in an appropriate file for future recall
- display the results as either a line graph or bar chart. The data for this display would have to be extracted from a derived data type, such as an array
- allow the user to interrogate the data to find information, such as the year of maximum carnivore population or the herbivore population for a specific year.

Design and develop an algorithm that:

- employs problem parsing to do a requirement analysis of the problem to identify the required classes and objects
- provides a description of the relationship that exists among the objects
- provides a generalized description of the methods and properties of each class
- uses a top-down, step-wise refinement approach to add specificity to each class
- uses an appropriate nomenclature, such as CRC Cards and Object Diagrams to outline the algorithm
- sketches a screen display(s) that illustrates the user interface and the format of the graphical output
- passes a walk-through or failed-on-paper test.

Translate the algorithm into an executable program that:

- maintains the logic and structure of the algorithm
- employs good object-oriented programming practices—loosely coupled, highly cohesive objects employing appropriate levels of encapsulation and data hiding
- incorporates the user interface and displays designed in the analysis and design stages
- uses a technique, such as iterative prototyping or parallel/recursive approaches to create and expand the core of functional code
- uses a combination of top-down coding and bottom-up coding:
 - to create the interface between individual objects
 - to create the objects
 - to test the objects
- makes appropriate use of derived data types to input, process and output information
- makes appropriate use of files to warehouse information
- has adequate internal and external documentation.

Test, implement and maintain the program by:

- executing the program with data known to produce specific output
- executing the program with aberrant data
- checking for congruency with the original requirements of the problem
- beta testing the program with knowledgeable users, such as another class member
- providing adequate installation, configuring and execution instructions to the end-user
- incorporating some user feedback in a revision of the program.

For assessment standards and criteria, see Assessment Checklist: Computer Science 3, INF3210-1.

Section I: Learning Resource Guide

NOTICE

Effective September 2002, Section I has been removed from all CTS strands and replaced with this general information page.

Alberta Learning authorizes a variety of resources that support learning and teaching in this strand. Teachers are encouraged to browse the Alberta Learning Web site at <http://www.learning.gov.ab.ca> on a regular basis for the most up-to-date information on:

- authorized resources; i.e., student basic, support and authorized teaching
- provincial software licensing agreements
- additional sources of support.

The lists of authorized resources that were previously included in Section I of the *1997 Guides to Standards and Implementation* have been deleted. Up-to-date listings of authorized resources are available at the Alberta Learning Web site and can be accessed through:

- Authorized Resources Database, a searchable online index of every approved learning and teaching resource for use in each subject area. The database is searchable for each 1-credit course.
- Learning Resources Centre (LRC). The LRC ensures accessible, available and affordable resources to enhance learning to all Alberta students.

A variety of documents and related sites are also accessible at the Alberta Learning Web site. These include:

- *Connection: Information for Teachers*, an online information newsletter for administrators, counsellors and teachers. It includes information on curriculum, resources, assessment, technology, new initiatives and projects.
- Learning Technologies Branch, a partnering branch that develops and provides information about distance learning programs and other learning alternatives offered by Alberta Learning.
- 2Learn Alliance, an education–business partnership that provides Internet inservice, support and mentorship for Alberta teachers.

